Techne: A Family of Multityped Formalisms for the Resolution of Requirements Problems

Jureta, Borgida, Mylopoulos

June 2, 2011

Abstract

A family of mathematical formalisms, called Techne, is introduced for the definition of requirements problems and the design, comparison, and ranking of their solutions. Starting from a very simple formalism, features of every subsequent formalism are incrementally added, motivated, illustrated, syntax and semantics are defined, and relationships to past work reviewed. It is argued that, and illustrated how Techne integrates many important ideas from past research on formalisms for modeling and reasoning in requirements engineering. Techne formalisms neither originate in, nor are specific to the engineering of software systems.

Contents

1 Introduction

The *requirements problem* states the purpose to be achieved and the constraints to be maintained by a system-to-be. A *solution* defines how the system-to-be shall do so and under what assumptions.

Central aims of requirements engineering (RE) as a field of research include making and evaluating ontologies, formalisms, and methodologies for the acquisition, interpretation, clarification, organization, archival, analysis, and validation of information needed to define the requirements problem and its alternative solutions, and to compare and rank these alternatives. Once one solution is chosen, several subsequent activities can begin, including software engineering, to produce the detailed specification of the solution's automated parts as a blueprint for the production and/or selection of the software parts of the system-to-be.

Once a requirements problem reflects issues in the interaction and coordination of people and machines, solutions will rarely be reducible only to software. While this clearly places RE outside software engineering, RE has been struggling to produce formalisms which both follow best practices in notation engineering *and* are designed specifically for the structuring of the requirements problem and solution space. This is best reflected in oft-cited formalisms for RE which have been constructed by reuse of formal methods in software engineering.

This paper introduces a family of formalisms, called *Techne*, designed from grounds-up specifically for the representation and reasoning about requirements problems, alternative solutions, and for the comparison and ranking of alternatives. Techne departs in several respects from prior research on the design of formalisms in RE and from formal methods in software engineering:

1. *Propositional language.* In every Techne formalism, *the smallest unit of information is a natural language proposition* so that the language of every Techne formalisms is propositional. This is in contrast to a (ground) predicate over object instances as in formal methods for software engineering [7] and in various oft-cited formalisms for RE (e.g., [9, 11]). Reasons for this design decision include the following:

    - *Modeling workload:* Natural language information obtained from stakeholders may suggest many alternative solutions, each of which needs to be defined, compared, and ranked. (E.g., consider how many

various ways can be imagined over a brainstorming session to secure access to an e-banking system, and note how there is no single way that is being adopted from one bank and country to another.) Many alternatives will end up being rejected after they are compared over various feasibility, usability, marketing, or other criteria. It seems very hard to justify the resource and expertise investment needed to define each of these alternatives in a predicate formalism.

- *Software reuse:* The general principle that reuse is a good practice means that the requirements problem and solution space will often be constrained by the knowledge of functionalities in available solutions, even if those solutions were intended for different requirements problems. Descriptions of existing solutions tend to be in given natural language, and the existence of the solution suggest no need (unless reengineering is an aim) to use a predicate formalism and sophisticated refinement operations thereon.
- *Uncertainty:* Alternative solutions refer to conditions in the future. It is difficult to be precise about such conditions. The more one wants to be precise, the more the expressive power of predicate formalisms becomes relevant. Using a predicate formalism to structure the problem and solution space thus means wanting to be precise about alternatives which are only roughly known.
- *Decision postponement:* The more expressive a formalism, the more decisions it expects from the modeler. E.g., using a temporal logic requires that decisions are made about the temporal relations between conditions referred to in statements. A predicate formalism asks one to decide about the variables in the domain, and therby about classes of objects in it. Choosing appropriate classes and their relations for each alternative solution does not seem relevant given that one solution will be chosen.
- *Decision authority:* Stakeholders of the system-to-be choose the solution. In order to understand alternatives and make their decision, the solutions ought to be described to them in a formalism where the language remains readable. It is not reasonable to assume that training in using predicate formalism can always be done, or that it will always be accepted, even if seemingly necessary.

Techne recognizes that the common form of information about requirements problems and solutions are natural language statements. An aim in making Techne was to show the variety of relevant analyses that can be performed over propositions that remain intuitively understandable to various stakeholders, before (and regardless of whether) these statements are rewritten as formulae over predicates in a software specification.

2. *Types.* Every Techne formalism has a set of special expressions which influence the roles that propositions can take in relations to other propositions and within the requirments problem and solution space. Each such

expression gives a *type* over propositions. E.g., the expression "it is a goal that p", where p is a proposition, is stated by adding the superscript G and writing p^{G} in various Techne formalisms, which reads "it is a goal that proposition p".[1]

3. *Multiple orthogonal types.* Some Techne formalisms have several sets of types, such as the sets of *core* (e.g., "it is a goal that", "it is a task to"), *optionality* (e.g., "it is mandatory that", "it is preferred that"), and *agency* types (e.g., "it is role x's responsibility to"). These different sets of types are orthogonal, in that no combination of them is explicitly forbidden, although some are discouraged.

4. *Formality.* A Techne *formalism* is (i) a formal *(multi)typed formalism* together with (ii) a language to write criteria for the comparison of, and decision rules for the ranking of alternative solutions to requirements problems, (iii) a *database* of expressions of these languages, and (iv) an *interface to the database* used to add to, remove from, and otherwise modify the database. By formalism is meant a formal language and a set of associated tools to manipulate its sentences, all of which are explained later on (cf., §3).

5. *Different formalisms for different needs.* The progression from the simplest Techne formalism to more complex ones is *a progression from a small to a large set of types on, and relations between propositions.* Starting from the simplest Techne formalism, every more complicated one is defined by adding features to a simpler formalism. Apart from facilitating presentation and discussion, this serves to show that each Techne formalism has its merits as a standalone formalism: the decision on which to use will depend on the tradeoff between the amount of information to enter into the database and the variety of questions that one wants the database to answer.

6. *Different problem and solution concepts.* Every Techne formalism comes with (i) a definition of the requirements problem that can be specified and resolved with that formalism, and (ii) the list of properties that every candidate solution specified in that formalism should satisfy. The progression from one formalism to the next is a progression from less to more detailed problem and solution concepts. The progression illustrates how simpler formalisms obscure information that other formalisms allow to be specified and analyzed. E.g., some formalisms cannot show agents' responsibilities over tasks, goals, or otherwise, while these same concerns can be specified in other Techne formalisms.

Starting from the simplest Techne formalism, features of every subsequent

[1] The informal reading of types in Techne may suggest that they resemble modalities. They are, however, not modalities in the same sense as necessity and possibility are modalities in modal logic, because, among others, nesting is not allowed in Techne (e.g., there is no Techne formalism which formalizes the statement "it is a goal that a task p").

formalism are incrementally added, motivated, illustrated, syntax and semantics are defined, and relationships to past work reviewed.

It is shown in this paper that various Techne formalisms synthesize in a coherent and precise way various ideas introduced separately in prior work on specification languages for requirements problems and solutions, including:

- concepts, such as *goal*, *softgoal*, *quality requirement*, *domain assumption*, *task*, *agent*, *role*, *plan*, *roadmap*, *adaptation requirement*;
- binary or n-ary relations, including *refinement*, *means-ends*, *decomposition*, *approximation*, *conflict*, *obstruction*, *preference*, *dependency*, *contribution*, *commitment*, *relaxation*, *responsibility*, *ability*;
- unary relations, such as *mandatory*, *preferred*; and
- rules of what is an appropriate refinement of some requirement, what is an appropriate approximation of a quality requirement, and so on.

Four aims are pursued in making and presenting Techne:

- *Theory building:* Techne aims to help establish the theoretical bases of formalisms for the representation of, and reasoning about requirements problems and solutions, or in other words, *a theory of knowledge representation and reasoning in requirements engineering.* The very basic and crucial departure from related work, which is necessary towards this aim, is no longer to *combine* contributions from other fields without significantly specializing them to specification of requirements problems and solutions. Instead, every Techne formalism is designed in such a way that everything in the formalism is closely integrated: the syntax, semantic domain, and semantic mapping, the structure of the database, the constraints on changes of the database, and the behavior of the database interface are all defined to reflect the basic intuitions suggested and developed over time in requirements engineering.
- *Synthesis & simplification:* An important result of Techne is that it carries over to a propositional formalism many important prior contributions made in the context of predicate logic-based formalisms for RE and software engineering. This results in very precise distinctions between *primitive* concepts and relations, and *derived* concepts and relations which are are *defined through combinations of primitives.* It is hoped that this simplifies, to colleagues and practitioners, the reading, use, and reviewing of past and future research on, and the teaching of the structuring and analysis of requirements problem and solution spaces.
- *Study & bridging of the informal/formal gap:* Natural language propositions are the smallest unit of information in every Techne formalism. Moreover, every Techne formalism is defined by incrementally adding features to a simpler Techne formalism. These two design principles make Techne into a study of how to transform information about requirements problems

and solutions from natural language statements to, as needed, increasingly structured (with every more complex Techne formalism) representations, until software specification by formal methods can take over.

- *Formalism design:* Design choices and tradeoffs made in designing every Techne formalism are discussed, to facilitate the reading of this paper as a case study in the design of formalisms. Increasing specialization of knowledge and systems favors a future in which no single (family of) formalisms will be judged adequate for most problems, making specific formalisms less interesting than *the principles to apply when making formalisms.* Techne is an illustration of some such principles.

This paper has two parts. The first part goes over preliminaries: (i) design principles and assumptions that were followed in designing every Techne formalism are discussed and contrasted to related research (§2); general components of, and their roles in a formalism are presented (§3); a running example is introduced (§4). The second part of the paper presents several Techne formalisms, T1 (§5)–T5 (§9), each having the components summarized in Table 1. The paper ends with a case study where Techne is applied.

2 Design Principles & Related Work

Design principles that were followed to make Techne resulted in departures from considerable prior research on specification languages and specification organization on requirements problems and solutions. They are discussed in this section. Related work is revisited in more detail when every Techne formalism is presented, in the second part of the paper. The reader interested only in the detail of the Techne formalisms can safely skip this section.

2.1 Design the Formalism to Fit the Problem Structure

A specification language should – by design – reflect assumptions about the structure of requirements problem and their solutions. There are two lines of research in which this is *not* the case. To see why/how, consider this very simple idea: to think about the system-to-be, it is useful to distinguish what is expected of it, from how it will deliver it. Say that Y somehow says the former, and X the latter. One would consequently want to check at some point if the following holds:

$$\text{Whenever } X, \text{ then also } Y. \tag{2.1}$$

It is necessary to check this in the engineering of *any* system-to-be: it is thereby a piece of knowledge that should be part of a formalism for RE.

Formal methods such as VDM [6], Z [30], Larch [14], or B [1] incorporate mechanisms to decompose the requirements problem into subproblems, then recompose solutions to subproblems into a solution of the original problem. While certainly very relevant, these mechanisms (e.g., schemas and ways to combine them in Z, traits in Larch) are *orthogonal* to, or equivalently, independent from,

Table 1: Overview of the Techne formalisms according to their components; Symbol "•" reads "present", "○" reads "absent".

Components		Formalisms				
		T1 (§5)	**T2** (§6)	**T3** (§7)	**T4** (§8)	**T5** (§9)
Core modalities	Goal	•	•	•	•	•
	Domain assumption	•	•	•	•	•
	Task	•	•	•	•	•
	Quality constraint	○	○	○	○	•
	Softgoal	○	○	•	•	•
	Soft dom. assumpt.	○	○	•	•	•
Optionality modalities	Mandatory	○	○	○	•	•
	Preferred	○	○	○	•	•
	Inherited	○	○	○	•	•
Agency modalities	Agent	○	•	•	•	•
	Role	○	•	•	•	•
Primitive relations	Inference	•	•	•	•	•
	Conflict	•	•	•	•	•
	Responsibility	○	•	•	•	•
	Ability	○	•	•	•	•
	Occupancy	○	•	•	•	•
	Commitment	○	•	•	•	•
	Preference	○	○	•	•	•
Variables	Binary Boolean	•	•	•	•	•
	Rational	○	○	○	○	•

any conceptualization of requirements problems, and therefore independent from our distinction between X and Y above. In other words, while a formal method *can* be used to check a condition such as Eq.2.1, it does not say that one *should* check such a condition.

That formal methods aim for the rigorous verification of software behavior is an advantage when one is interested in designing and verifying a software program. This same characteristic makes them difficult to apply when designing requirements problems and their solutions:

- Requirements problems often involve vague, incomplete, ambiguous, conflicting, and/or otherwise problematic goals, assumptions, and practices of people, and it is simply neither feasible, nor often relevant to convert statements about these into formulae over predicates before these statements have been clarified, and after it has been established what will be the responsibilities of software and what of people using the software.

- Requirements need to be validated by stakeholders, which in turn normally requires that stakeholders can understand a specification of the requirements problem and one or more of its candidate solutions. Doubts about

the feasibility of this were noted before (cf., e.g., Stidolph & Whitehead [31] for a discussion and relevant references).

- When goals and assumptions are unclear, there can be many candidate solutions to the same requirements problem. The exploration of alternatives and their comparison over criteria such as cost and usability do not justify every solution to be specified using a formal method (e.g., usability often involves some form of graphical user interface elaboration, which often uses ad-hoc visual syntaxes – cf., e.g., Bäumer et al. [5] for an early survey).

These are not arguments against formal methods themselves, but *against their use for tasks to which they simply do not apply very well*. Increased reliance on software, their long lifetimes, and increasing regulation thereon requires systematic engineering, where formal methods have a clear role (cf., e.g., [19]).

To properly position the arguments above, it is important to clarify a confusion of terminology between the formal methods and requirements engineering (RE) communities. This confusion may mislead to believe that formal methods can cover all steps, from the unclear statements up to a detailed software specification. In RE, requirements can be anything from the organizational (business) or individual goals, assumptions, and practices, however unclear they are, up to and including fragments of a software specification written using a formal method. In formal methods, however, the term *requirement* is interpreted in a more restricted way. Namely, there is an implicit distinction between "high-level objectives of an enterprise" and requirements, as suggested by Jones, Till & Wrightson [17]. Van Lamsweerde, Darimont & Letier [32, p.910] explicitly distinguish the *goal* and *requirement* concepts: "a requisite is a goal that can be formulated in terms of states controllable by some individual agent [...] A requirement is a requisite assigned to a software agent". More recently, the survey of practice and experience of formal methods application, from Woodcock et al. [36], mentions projects in which goals, assumptions, and practices relevant to the systems have already been distilled from high-level goals, in part due to experience with similar-purpose past systems (survey mentions microprocessor chip design, railway signaling and train control, smartcards for low-value cash-like transactions, flight control computers, barriers in seaports, access control).

RE did recognize these drawbacks at least since the 1980s, which was one of the motivations to create new specification languages, usually called *requirements modeling languages* (RMLs). That there is more to writing requirements than functional specification was recognized in the original RML [13] (hereafter ORML), "a notation for requirements modeling which combines object-orientation and organization, with an assertional sublanguage used to specify constraints and deductive rules" [12]. Formal semantics is given to ORML via a mapping from its descriptions to assertions in a predicate logic. One thereby obtains facilities for the structuring and organization of predicate logic theories. The ontology in ORML distinguished between entities, activities, and assertions. This was judged limited and responses to limitations went in two directions. RMLs such as KAOS and i^* took the direction in which the ontology remains fixed (i.e., one cannot add or remove concepts when applying the RML) but include more concepts,

designed to distinguish goals, assumptions, tasks, and so on. The other direction was adopted in Telos [22] and consists of leaving the ontology undefined, while having in the language the facilities needed to define an ontology. The second approach is more expressive, but its abstraction makes it difficult to provide methodological guidance which can be given when a fixed set of concepts is known and manipulated every time the language is used.

An important and subtle limitation of RMLs is that they keep these distinctions *outside of* the mathematics of the specification language. To make this clearer, consider how KAOS [9] is designed:

> "The overall approach taken in KAOS has three components: (i) a conceptual model for acquiring and structuring requirements models, with an associated acquisition language, (ii) a set of strategies for elaborating requirements models in this framework, and (iii) an automated assistant to provide guidance in the acquisition process according to such strategies." [9]

Of interest here is the first point, where a conceptual model is combined with an acquisition language. The acquisition language is a discrete-time and first-order linear temporal logic, which departs from languages in formal methods mainly by the absence of structuring mechanisms (e.g., *traits* and their relations in Larch, *schemas* and their relations in Z). The conceptual model provides the structuring mechanism, by defining an ontology over a number of concepts (object, operation, agent, goal, obstacle, requisite, requirement, assumption, scenario) and relations (specialization, refinement, conflict, operationalization, concern, and so on) [9, 32, 33]. Using the refinement relation, goals are organized into AND/OR trees, and each goal can carry a theory (i.e., set of formulae) written in the acquisition language.

The design principle in KAOS is to define an ontology and have theories in a logic correspond to instances of the concepts from the ontology. This principle was followed subsequently in RE when the aim was to make specification languages that can go all the way from loose goals to formal method-like specifications (e.g., Formal Tropos [11]). The principle has two important drawbacks, both of which arise from keeping the conceptual model outside of the mathematics in the specification language, or in other words, defining the mathematical logic and the ontology independently from one another:

- When a formula or set of formulae is an instance of a concept[2], such as a goal, then a goal corresponds to a pattern of formulae. The patterns are: $p \Rightarrow \Diamond q$ is called *achieve goal* ("if p, then eventually q"); $p \Rightarrow \Diamond \neg q$ a *cease goal* ("if p, then eventually not q"); $p \Rightarrow \Box q$ a *maintain goal* ("if p, then always q"); and $p \Rightarrow \Box \neg q$ a *avoid goal* ("if p, then never q"). Every

[2]There is more to a goal in KAOS and Formal Tropos than only formulae. There is a natural language statement that was rewritten in formulae, and there is other bookkeeping information. Their presence does not, however, affect the argument that is given, since the argument is not about the information in the goal, but what the goal is in a mathematical logic used to write formulae.

instance of an achieve or cease goal thus gives a *liveness property* to be satisfied by the system, while an instance of maintain or avoid goal gives a *safety property*. All of these goal types are independent from the logic in patterns: one could replace the logic, and not much would change (as long as it has negation, implication, and the temporal operators used in the patterns). One could also replace the patterns, and nothing would change in the ontology. This suggests that there is little influence of the ontology, that is, of the conceptualization of the requirements problems and solutions, on the logic that constrains the representation and reasoning about these problems and solutions.[3] In contrast, in every Techne formalism, the ontology is a source of modalities over atomic facts in the mathematical logic, so that a goal is a proposition with a modality, not a pattern of formulae. The modalities are part of the logic.

- When an ontology for requirements is used as a structuring mechanism, concepts useful for thinking about requirements are confused with those relevant for structuring the requirements. In other words, a concept such as *goal* should not play the role of a *schema* (as in Z) or *trait* (as in Larch), with refinement playing the role of inclusion relations in formal methods. A goal serves to say what is desirable, not that it is part of something else. The point is that *an ontology for requirements should be orthogonal to an ontology of concepts used for the structuring of requirements*. Principles for the structuring of specifications are independent from our preference of goals and tasks, over states and transitions, hence that orthogonality.

In a summary, this first design principle for Techne has the effect that an existing logic is not combined with an ontology for requirements. Rather, a logic is made to fit the ontology for requirements.

2.2 Refine the Informal/Formal Distinction

A systems development project can be completed without the application of formal methods, but not without statements of requirements in natural language. For projects that do not have the maturity and criticality of those surveyed by Woodcock et al. [36], surveys of practice in industry suggest that "formal" representation of requirements problems and solutions is rare – e.g., Neill & Laplante report 7% of their sample of practitioners used some such representation [24]. Although it remains unclear what is meant exactly by "formal" and "informal", it is reasonable to assume that the limits of this continuum are solutions written using a specification language from a formal method on the formal side, and natural language documents on the informal side.

Instead of suggesting a single formalism with which to go from informal statements of problems and solutions to expressions in formal methods, the

[3] This independence also begs the question of what drawbacks were accepted in exchange for reusing an existing logic. When a discrete-time and first-order linear temporal logic is adopted, two obvious ones are trivial deductions if requirements are inconsistent, and persistence of past deductions when new requirements are added. This is discussed further below.

aim with Techne is to have various formalisms which can be used to decompose informal documents onto simpler natural language propositions, relate these propositions, and ask questions about these relationships.

If one is interested in requirements for an ambulance dispatching system, a statement in an informal document, such as *If an ambulance is dispatched to the incident location and the ambulance confirms that it has arrived at incident location, then the incident is handled*, will be converted in a Techne formalism into $\mathbf{g}(p_1) \wedge \mathbf{g}(p_2) \rightarrow \mathbf{g}(q)$, where $\mathbf{g}$ is a and goal modality, read "it is a goal that", while p_1 refers to *An ambulance is dispatched to the incident location*, p_2 to *Ambulance confirmed arrival to incident location*, and q to *Incident is handled*.

To go from a specification in Techne to a specification in a predicate logic, in a formal method, consists of rewriting individual natural language propositions as formulae of predicate logic, and mapping relations from the Techne formalism onto patterns of formulae in the predicate formalism. This is not studied in this paper.

2.3 Tolerate Inconsistency

Let Δ be the set of formulae referring to information about a requirements problem and its solutions, i.e., a *requirements database*, written in a logic named x. It does not matter what specific logic x is, only some properties assumed for it below.

When a specification language uses a variant of propositional or predicate *classical* logic, its consequence relation $\vdash_x$ is defined by a proof theory which includes the *ex falso quodlibet* (EFQ) proof rule [27]. EFQ concludes anything from an inconsistency. Let $\mathcal{L}$ be the set of *all* formulas of the logic x; EFQ is:

$$\forall \alpha \in \mathcal{L},\ \frac{\bot}{\alpha}. \qquad \text{(EFQ)}$$

When EFQ is allowed in the proof theory of x, deduction from an inconsistent Δ returns useless conclusions. E.g., every requirement is a consequence of an inconsistent Δ, which would lead to the erroneous conclusion that an inconsistent Δ satisfies every requirement. The relation $\vdash_x$ is called *trivializable* (TRIV) [15] when it behaves so, or equivalently:

$$\text{If } \Delta \subset \mathcal{L} \text{ and } \Delta \vdash_x \bot, \text{ then } \forall \phi \in \mathcal{L},\ \Delta \vdash_x \phi. \qquad \text{(TRIV)}$$

In order to avoid clearly useless conclusions from an inconsistent Δ, a specification language must: (a) have a consequence relation $\vdash_x$ that *fails* TRIV, and thereby is *paraconsistent*, or (b) ask its user to resolve all inconsistencies in Δ, thus obtain a set of formulas, denote it $R(\Delta)$, which is consistent, i.e., $R(\Delta) \nvdash_x \bot$, and then draw conclusions from $R(\Delta)$.

Specification languages for requirements, those which do include a mathematical logic, use a variant of classical logic, so that their $\vdash_x$ succeeds at TRIV. These RMLs (cf., Robinson, Pawlowski & Volkov's survey [26], and later work, e.g., [11]) consequently follow the second approach (b) and offer strategies for

the identification and resolution of inconsistencies in Δ in order to obtain a consistent $R(\Delta)$ (e.g., [32, 33]).

In contrast to such *classical* languages, which succeed at TRIV, a *paraconsistent* formalism (i.e., with a non-trivializable specification language) would allow non-trivial conclusions to be drawn from an inconsistent Δ *before* any change thereto is made towards a consistent $R(\Delta)$.

To see more clearly why this makes paraconsistent specification languages more interesting than classical ones, consider the range of questions that one can ask a specification language, the answers being logical consequences of a set of well-formed formulas in the logic of the specification language. Let $\vdash_c$ be the consequence relation of a classical specification language and $\vdash_p$ that of a paraconsistent specification language. Since $\vdash_c$ succeeds at TRIV, the following is obvious:

$$\text{If } \Delta \vdash_c \bot, \text{ then } \forall R(\Delta) \text{ s.t. } \Delta \subseteq R(\Delta),\ R(\Delta) \vdash_c \bot. \tag{2.2}$$

Equation 2.2 says that if Δ is inconsistent, then it is impossible (if using a formalism with $\vdash_c$) to make a consistent $R(\Delta)$ only by *expansion*, i.e., only by *adding formulas to* Δ. This is a simple, but important observation, because from it follows that two options remain to make a consistent $R(\Delta)$ out of Δ: *revision* or *contraction*. Revision consists both of adding new formulas to Δ and removing some formulas from Δ in order to make a consistent $R(\Delta)$. Contraction consists of only removing formulas from Δ in order to make $R(\Delta)$. Since $R(\Delta)$ must be obtained by revision or contraction of Δ, this is obvious:

$$\text{If } \Delta \vdash_c \bot \text{ and } R(\Delta) \nvdash_c \bot, \text{ then } \Delta \neq R(\Delta). \tag{2.3}$$

According to Equation 2.3, a classical specification language must answer questions, i.e., derive conclusions, not from the inconsistent database of requirements Δ, but from an $R(\Delta)$, obtained by changing Δ. Among others, two kinds of questions are of interest in RE:

1. *What are the logical consequences of* $X \subseteq Y$*?* where Y is a (part of) a database of requirements, which for a classical specification language must be consistent. In RE, it is relevant to ask which requirements are satisfied by some subset of requirements.

2. *Can* z *be derived from* $X \subseteq Y$*?* where Y is (part of) a database of requirements, and z is either a formula or $z = \bot$. This is asked RE in order to know whether a requirement $z = \phi$ can be derived from (and this is usually interpreted as "is satisfied by") a set of requirements.

The range of questions to ask on a given requirements description Δ can be characterized through the set of all answers that can be obtained: the closure of Δ is the set of all logical consequences of Δ and thereby of all possible answers that can be obtained from Δ in a given logic. The closure of a set of formulas Y in a logic with $\vdash_x$ is $Cl(\vdash_x, Y) = \{\phi \mid Y \vdash_x \phi\}$. Observe from TRIV above that the $\vdash_c$ -closure of an inconsistent Δ is the set of all formulas in the logic of

the classical specification language with $\vdash_c$, i.e., $Cl(\vdash_c, \Delta) = \mathcal{L}_c$, leading to the following:

$$\text{If } \Delta \vdash_c \bot, \text{ then } Cl(\vdash_p, \Delta) \subset Cl(\vdash_c, \Delta) \tag{2.4}$$

The set $Cl(\vdash_c, \Delta) \setminus Cl(\vdash_p, \Delta)$ includes all consequences of Δ that cannot be obtained using $\vdash_p$.

The following observation is important. It comes from Equation 2.3, while assuming that Δ is inconsistent and $R(\Delta)$ is consistent and obtained by revising or contracting Δ:

$$Cl(\vdash_p, \Delta) \neq Cl(\vdash_p, R(\Delta)) \tag{2.5}$$

That there is a difference between $Cl(\vdash_p, \Delta)$ and $Cl(\vdash_p, R(\Delta))$ suggests that *the same questions cannot be asked to* Δ *and to* $R(\Delta)$*, and thus that questions asked on* Δ *using* $\vdash_p$ *cannot be asked on* $R(\Delta)$ *using the same* $\vdash_p$. I.e., some questions and their answers are lost in the revision or contraction that changes Δ into $R(\Delta)$.

Outside RE, a related argument in favor of paraconsistency has been formulated as follows (e.g., [15]). A paraconsistent logic lets one avoid making *premature* decisions to restore consistency to an inconsistent set Δ of formulas. Instead of deciding how to eliminate inconsistency every time it is encountered, one draws conclusions while tolerating inconsistency in Δ, and thereby avoids the loss of information observed above in Equation 2.5. In the context of RE, this amounts to say that one need not resolve an inconsistency as soon as it is detected, but can let it be and, as is the case in Techne formalisms, make use of it to decide which alternative set of requirements the system should be engineered to satisfy.

Overall, a paraconsistent logic defined over a specification language would allow useful conclusions to be drawn from an inconsistent description of requirements Δ, or, stated otherwise, be able to answer questions of methodological interest in RE.

3 What is a Formalism?

The aim in this section is to overview the general components found in Techne formalisms, and relate the notion of *formalism* to those of *syntax*, *semantics*, *formal (specification) language*, *logic*, *database* and *interface*.

3.1 Methodological Questions

The aim of a formalism is representation of information and automated reasoning thereon. It is useful as long as it can be used to answer questions of interest with respect to the problem one wishes to solve. When working with requirements problems and solutions, the broad questions of interest are as follows:

Q1: *What kinds of information are relevant?* The formalism should identify the kinds of information to elicit from the stakeholders of the system-to-be, and the kinds of information that may not necessarily be amenable to elicitation, but is still relevant when dealing with problems and solutions in RE.

Q2: *What are the aims that the information serves?* The overall aim is to understand a particular requirements problem and find solutions to it. The formalism should define precisely the requirements problem and the candidate solution concepts.

Q3: *How to explicitly represent information?* Instances of the various relevant kinds of information need to be represented and recorded in some format amenable to communication and analysis.

Q4: *What conclusions to draw from explicit information?* Explicitly represented information may allow various informal, potentially ambiguous, conclusions. The formalism should suggest which conclusions are correct.

Q5: *How to transform information to achieve those aims?* Elicited information needs to be clarified, made more detailed, and otherwise modified to formulate the specific requirements problem and its candidate solutions. The formalism should say how to transform information towards these aims.

Q6: *How to verify if the aims have been achieved?* Determining if a specification is effectively a solution to a specific requirements problem requires drawing conclusions about the relationships between the former and the latter.

Q7: *How to change the representation of information?* Requirements may change as a result of learning more about the requirements problem and alternative ways of solving the problem. E.g., this can be the case when it is necessary to reestablish the consistency of some subset of requirements. The formalism should suggest what changes are allowed on a representation of information and what the consequences of these changes are.

Q8: *How to compare and rank alternatives?* There are likely to be various alternative solutions to the requirements problem. The formalism should say which information gives criteria for the comparison of candidate solutions, and how these criteria are used to rank candidates.

The formalism that one chooses should, by design, suggest how to answer some or all of these questions. The questions are intertwined: e.g., deciding which conclusions are correct and determining if a specification is a solution to a requirements problem both concern the rules of inference in the formalism. This suggests that there are limitations, as discussed earlier (§2) to combining components from general-purpose formalisms and applying them to RE. The rest of this section discusses various components and their relationships to the questions above.

3.2 Explicit Information and Its Classification

Syntax is a set of *symbols* combined with a *grammar*. Combinations of symbols allowed by the grammar are usually and interchangeably called expressions, sentences, or (well-formed) formulas. Symbols refer to objects in a semantic domain. The semantic mapping function says which symbol or combination of symbols refers to, respectively which object or combination of objects in the semantic domain.

The combination of syntax, semantic domain, and semantic mapping function is a *formal language*, and if used for the specification, a formal specification language. This is usual terminology, shared by, e.g., Wing [35].

A convenient way to describe a semantic domain is to define an ontology of the information judged relevant to the problem that is being studied. The semantic domain is then a set of instances of the concepts in the ontology, or equivalently, the union of all extensions of all concepts in the ontology. A relation between two concepts in the ontology refers to a relation between the objects in the extensions of the respective concepts.

The concepts and relations in the ontology are normally reflected in the syntax of the formal language. Concepts may define sets of symbols, each corresponding to a member of the extension of the concept, while relations may be referred to by connective symbols. If a formal language where natural language propositions are members of the semantic domain, and where conjunction is a relation between some propositions, the syntax is likely to include symbols, say, p, q, r, indexed as needed, to refer to propositions and a connective symbol, e.g., $\wedge$ to refer to the conjunction relation, if the formal language is part of a formal logic. Grammar will restrict how propositions and connective symbols can be combined.

The combination of a semantic domain and a semantic mapping function, as they are described above, can be called *domain semantics*, to distinguish it from other and compatible semantic domains and semantic mapping functions (e.g., correspondence semantics discussed below). A formal specification language needs at least syntax and domain semantics.

Example 3.1. (A simple formal language.) Suppose that the ontology has only one primitive concept, called *requirement*, and that its instance is any natural language proposition elicited from the stakeholders of a system-to-be. There are two relations in the ontology, in which instances of X can stand, namely conjunction and conditional ("if..., then...") relation. Let the set of symbols for propositions be a finite set $\{p_1, \ldots\}$, and p be some generic member of that set. Let the grammar be defined in Backus-Naur Form (BNF) as follows:

$$\phi ::= p \mid p_1 \wedge \ldots \wedge p_{n \geq 1} \rightarrow p_{n+1} \tag{3.6}$$

where ϕ is an expression, and the symbols $\wedge$ and $\rightarrow$ refer to, respectively, the conjunction and conditional relations. Informally, a specification in this language can be read as an AND/OR tree of requirements, each written as a natural language proposition, where an AND node is any proposition p that is on the right-hand side of $\rightarrow$, while p is an OR node if there are two or more expressions where it is on the right-hand side of two or more expressions with $\rightarrow$. ■

Let $\mathcal{L}$ denote the set of all expressions that can be written using symbols and grammar in some formal language called x. Let P be the set of all propositions (not proposition symbols, but that which the symbols refer to), and $\mathcal{P} : \mathcal{L} \longrightarrow P$. The resulting formal language is tuple:

$$(\mathcal{L}, P, \mathcal{P}). \tag{FL}$$

Expressions written in a formal specification language can be called *explicit information*. The choice of domain semantics of the formal language answers Q1. The symbols and grammar together with domain semantics answer Q3. By defining relations between pieces of information, an FL also responds to Q5: in Example 3.1, AND/OR tree-like structures can be used to transform explicit information in order to add detail to it – the refinement relation between requirements is indeed often captured by AND/OR trees in RE.

3.3 Implicit Information and Its Derivation

Implicit information are the conclusions that can be drawn from the expressions in the formal language. Implicit information cannot be controlled through a formal specification language alone. Such control requires the definition of the rules to use to draw conclusions.

A *logic* is typically made of a formal language and semantics *other than* (and/or in addition to) domain semantics. Since Techne is a family of propositional formalisms, semantics typical for predicate languages are not discussed.

Proof-theoretic semantics amount to a definition of the roles that expressions can have in deductions. It requires the definition of a *proof theory*, that is, rules of inference and/or axioms which together define the consequence relation $\vdash_x$ of x. Such a logic is the tuple:

$$(\mathcal{L}, P, \mathcal{P}, \vdash_x). \tag{LOG}$$

Example 3.2. (A simple logic.) Continuing Example 3.1, suppose that the following inference rule is defined:

> *If* $p_1, \ldots, p_n$ *and* $(\bigwedge_{i=1}^{n} p_i) \rightarrow p_{n+1}$, *then conclude* p_{n+1}.

With no axioms, a consequence relation $\vdash_x$ can be defined from this proof rule (i.e., *modus ponens*). One possible definition is:

> Let $Y \subseteq \mathcal{L}$, and $\phi \in \mathcal{L}$. $Y \vdash_x \phi$ if and only if (i) $\phi \in Y$ or (ii) ϕ is such that $(\bigwedge_{i=1}^{n} p_i) \rightarrow \phi$ is in Y and for $1 \leq i \leq n$, $Y \vdash_x p_i$.

This consequence relation says that ϕ can be deduced from a set Y iff that set already includes ϕ, or if there is an implication that has p as its consequent, and the antecedents in the implication can all be deduced from Y. ■

Proof semantics are concerned with the correctness of deductions with regards to rules of inference and/or axioms. To state whether propositions and expressions over propositions correspond to observed conditions "in the world" requires

correspondence semantics. For simplicity, the semantic domain includes two "truth values", *true* and *false*. The semantic mapping function can be defined inductively for a given language as in the following example.

Example 3.3. Correspondence semantics for the language defined in Example 3.1 is a set $\{true, false\}$, and the semantic mapping function defined inductively as follows:

- Proposition p is true if and only if (iff) $\mathcal{P}(p)$ accurately describes the world;
- Expression $p_1 \wedge \ldots \wedge p_{n\geq 1} \rightarrow p_{n+1}$ is true iff if every p_i, $1 \leq i \leq n$ is true, then p_{n+1} is true.

■

The semantic mapping function in correspondence semantics can also be called a *model*, denoted $\mathcal{M}$, and is such that $\mathcal{M} : \mathcal{L} \longrightarrow \{true, false\}$. The satisfaction relation $\models$ can then be defined as follows:

$$\mathcal{M} \models \phi \text{ iff } \mathcal{M}(\phi) = true. \tag{3.7}$$

The satisfaction relation can be inductively defined by analogy to how truth values are assigned to propositions and expressions.

Example 3.4. Given correspondence semantics for $\mathcal{L}$, i.e., the pair $(\{true, false\}, \mathcal{M})$, the satisfaction relation $\models$ for x is defined inductively as follows:

- $\mathcal{M} \models p$ iff $\mathcal{M}(p) = true$;
- $\mathcal{M} \models p_1 \wedge \ldots \wedge p_{n\geq 1} \rightarrow p_{n+1}$ iff if $\mathcal{M}(p_i) = true$ for every proposition p_i, $1 \leq i \leq n$, then $\mathcal{M}(p_{n+1}) = true$.

■

The logic with correspondence semantics is the tuple:

$$(\mathcal{L}, P, \mathcal{P}, \vdash_x, \models). \tag{LOG+CS}$$

An LOG or LOG+CS responds to Q1, Q3, Q4, Q5, and Q6. By stating which conclusions are correct, it addresses Q4, and thereby also Q6.

Techne formalisms do *not* have correspondence semantics, only proof semantics. This still allows to talk about satisfaction in Techne, since $\vdash_x \phi$ can be read "ϕ is satisfied", without checking that also $\models \phi$.

3.4 Comparison & Ranking

Alternative refinements of requirements and alternative ways of satisfying them result in more than one candidate solution to a requirements problem. All alternative solutions will share the same properties, such as that each is internally consistent, but once there are more than one of them, it is necessary to have ways of representing and using information for their comparison and ranking.

A logic can be used to define the threshold properties that a set of expressions should have in order to count as a solution to a requirements problem. E.g., if one such property is consistency, then a set S will *not* be a candidate solution if $S \vdash_x \bot$. If consistency is the *only* necessary property, then S will be a candidate solution if $S \nvdash_x \bot$.

Specific threshold properties of solutions will be discussed later and for each Techne formalism. These properties form a *threshold*, in the sense that a set of information *must satisfy all these properties* to count as a solution. It follows that these properties are *not* criteria for the comparison of candidate solutions.

There are, broadly speaking, three kinds of criteria for the comparison of solutions:

- Criteria from binary preferences over requirements: e.g., a preference relation usually says that satisfying one requirement is strictly more desirable to satisfying another one. Each preference gives one criterion: a solution which satisfies the preferred requirement ranks higher, over that criterion alone, than any solution which satisfies the less preferred requirement.

- Criteria from individual preferred requirements: not all requirements must be satisfied by a candidate solution, some can be violated. For a requirement which can be violated, it is more desirable to satisfy it than to violate it. This corresponds to a special kind of preference relation, between a requirement and its negation, in contrast to preferences mentioned above. Every preferred requirement gives a corresponding criterion: over that criterion alone, a solution which satisfies the preferred requirement ranks higher than a solution which satisfies the negation of that requirement.

- Criteria from nonfunctional requirements: different solutions will result in different levels of reliability, performance, security, safety, usability, and so on. Such nonfunctional requirements give criteria alike binary preferences: if a system-to-be should be reliable, and there is measure of reliability, then this nonfunctional requirement gives a ranking of candidate solutions. Namely, candidates with higher values of the reliability measure are more desirable – over the reliability criterion only – than candidates with lower values of that same measure.

A set of criteria alone is an input to decision-making towards the selection of a solution. As the number of criteria and candidates increases, it becomes relevant to introduce *decision rules*. Each criterion provides a scale on which every candidate solution obtains a value. A decision rule is a function that returns, for a given solution, the rank of that solution over an aggregate of one, some, or all criteria. A decision rule may also suggest to choose the highest-ranking solution returned by that decision rule.

The combination of criteria with a logic may be done in two ways. One way is to allow preferences into the syntax, and have them act just like any other connective. The other way is to restrict more how preferences can be written using the logic. In both cases, (i) preference relations are relations between

members of the semantic domain in the domain semantics of the logic, and (ii) it should be decided how, if at all, preference relations have a role in the proof semantics and/or correspondence semantics of the logic.

Let $(\mathcal{L}, P, \mathcal{P}, \vdash_x)$ be a logic as discussed earlier in this section. To have criteria for comparison, add to the logic the preference relation and indifference relations, denoted, respectively $\succeq$ and $\approx$. Both are binary relations between expressions in $\mathcal{L}$, i.e., $\succeq \subseteq \mathcal{L} \times \mathcal{L}$ and $\approx \subseteq \mathcal{L} \times \mathcal{L}$. $\psi \succeq \phi$ is informally interpreted read "satisfying ψ is at least as desirable as satisfying ϕ". $\psi \approx \phi$ reads "satisfying ψ is as desirable as satisfying ϕ".

Every candidate solution is represented using the language of the logic, so that a candidate equates to a set of expressions. A decision rule, denoted $\mathcal{D}$, is then a function from a set of candidates to a partial order of these candidates, and the decision rules is a set of such functions ($\wp$ returns the powerset of its parameter):

$$\mathfrak{D} \stackrel{def}{=} \{\mathcal{D} \mid \mathcal{D} : \wp(\wp(\mathcal{L})) \longrightarrow \{\geq | \geq \subseteq \wp(\mathcal{L}) \times \wp(\mathcal{L})\}\} \qquad \text{(DR)}$$

Any particular decision rule is likely to be specified as a macro. Taking a set of candidate solutions as its input, the macro will apply rules for the aggregation of ranks of candidates over individual criteria. Its output is a partial order on the members of its input set. Loosely speaking, the partial order will say which candidates are better than others according to the rules in the macro.

Example 3.5. Continuing with previous examples, allow a strict preference relation $\succ$ between any pair of expressions. A decision rule may be the following macro, call it M:

1. Initiate the score of every candidate solution by setting it at 0.
2. For every candidate solution S, if a proposition that is strictly preferred in a preference relation is (i) in the candidate solution S, or (ii) can be deduced from S, then add 1 to the score of S.
3. Repeat Step 2 above until all preference relation pairs have been processed.
4. Define the binary relation $\geq_M$, such that if the score of S_i is equal or greater than the score of S_j, then $S_i \geq_M S_j$.

This macro outputs the relation $\geq_M$ over the candidate solutions given to it. It has the obvious drawback that each preferred proposition adds the same value (plus one) to the score of a candidate solution: i.e., all preferences are equally important. ■

Adding preference and indifference relations, and decision rules to LOG results in the tuple:

$$(\mathcal{L}, P, \mathcal{P}, \vdash_x, \succ, \approx, \mathfrak{D}) \qquad \text{(LOG+PREF+DR)}$$

Adding preference relations and decision rules to a logic gives a structure gives an answer to question Q8. Preferences allow the representation of criteria, while decision rules rank alternative solutions using these criteria.

3.5 Database & Interface

Given a formal language for requirements problems and solutions, a *requirements database* – denote it Δ – is simply a set of expressions in that language. Using the database for storage and retrieval requires an *interface* – denote it `I` – which includes operations on the database.

To interact with a knowledge base, Levesque & Lakemeyer's [21] use three primitive operations: (i) `ASK`, to retrieve from a knowledge base; (ii) `TELL`, to add to the knowledge base; and (iii) `INITIAL`, to retrieve the content of the knowledge base before any `TELL` operation was performed. In *Telos* [22], Mylopoulos, Borgida, Jarke & Koubarakis use the operations `TELL`, `UNTELL`, and `RETELL` to add or revise a knowledge base, while `RETRIEVE` and `ASK` serve to retrieve from it.

Two sets of primitive operations should be found in an interface:

- *Operations on explicit information* require a formal language. It should be possible to add, remove, and retrieve explicit information from the database.

- *Operations on implicit information* require a logic. These operations apply not on Δ, but its closure over the consequence relation in the logic: if $\vdash_x$, then the operations apply on $\bar{\Delta} \stackrel{def}{=} \{\phi \mid \Delta \vdash_x \phi\}$. Operations on $\bar{\Delta}$ should cover the common operations in belief revision, which are likely to include some forms of expansion (adding to $\bar{\Delta}$ an expression which is consistent with $\bar{\Delta}$), contraction (removing from $\bar{\Delta}$), and revision (adding information which makes $\bar{\Delta}$ inconsistent and then changing the expanded $\bar{\Delta}'$ to make it consistent).

Operations on explicit information can be obtained through, e.g., SQL if Δ is implemented as a relational database.

Operations on $\bar{\Delta}$ are more interesting. Let $\bar{\Delta}'$ be the result of applying some such operation on $\bar{\Delta}$. The definition of operations on $\bar{\Delta}$ will depend on the properties of the consequence relation *and* on the properties of $\bar{\Delta}$ that one wishes to always maintain.

For example, Alchourrón, Gärdenfors & Makinson (AGM) [2] gave definitions of operators when $\vdash_x$ is classical (and thus, not paraconsistent) *and* if $\bar{\Delta}$ *must* be consistent. In that case: (i) adding an expression ϕ which is consistent with $\bar{\Delta}$ results in the consistent $\bar{\Delta}' \stackrel{def}{=} \{\psi \mid \Delta \cup \{\phi\} \vdash_x \psi\}$; (ii) removing ϕ from $\bar{\Delta}$ requires one to decide which formulas to take out to ensure that $\bar{\Delta}' \nvdash_x \phi$; (iii) similarly to contraction, revision requires deciding how to change $\bar{\Delta}$ when $\bar{\Delta} \cup \{\phi\} \vdash_x \bot$, to obtain a consistent $\bar{\Delta}'$. Since it was argued that AGM operators are general, it would be expected that they apply to databases of requirements problems and solutions. It will be shown later that this is not the case (cf., §5).

A *formalism* is a structure obtained by adding the database and its functional interface for explicit and implicit information, and is the tuple:

$$(\mathcal{L}, P, \mathcal{P}, \vdash_x , \succ, \approx, \mathfrak{D}, \Delta, \mathtt{I}) \qquad \text{(FORMALISM)}$$

The tuple covers all methodological questions Q1–Q8.

4 Running Example

The examples used throughout the paper draws on the London Ambulance Service case study [3]. The requirements used in examples are for an ambulance dispatching system (ADS hereafter). Examples are presented in increasing detail, as each Techne formalism allows.

5 T1

T1 is the simplest Techne formalism. It can be used to specify goals, domain assumptions, and tasks, and the inference and conflict relations between them. The formalization of the inference relation is such that the fragment of the requirements database in **T1** that excludes conflict relations can be visualized as an AND/OR forest. The forest can be interpreted as showing refinement and decomposition of requirements, so that **T1** can be used to write the common "goal trees".

5.1 Illustration

A goal for an ADS is that ambulances arrive at incident locations. This is written in **T1** as a propositional symbol, e.g., q_1 which refers to the proposition "Ambulances arrive at their incident locations". The symbol is labeled **g** since the conditions stated in the proposition are desired. In short, $\mathbf{g}(q_1)$.

A basic feature needed in any specification language for requirements is a mechanism that handles the adding of details to requirements which have already been identified. Usually called refinement or decomposition, the mechanism is realized in **T1** using the inference relation.

The goal $\mathbf{g}(q_1)$ can be satisfied if all following goals are satisfied together:

$\mathbf{g}(p_1$: Identify available ambulances);

$\mathbf{g}(p_2$: Choose ambulance);

$\mathbf{g}(p_3$: Assign ambulances);

$\mathbf{g}(p_4$: Mobilize ambulances);

$\mathbf{g}(p_5$: Confirm mobilization).

The relationship between $\mathbf{g}(p_1), \ldots, \mathbf{g}(p_5)$ and $\mathbf{g}(q_1)$ is such that if $\mathbf{g}(p_1) \wedge \ldots \wedge \mathbf{g}(p_5)$ can be deduced, and $\mathbf{g}(p_1) \wedge \ldots \wedge \mathbf{g}(p_5) \rightarrow \mathbf{g}(q_1)$ can, then $\mathbf{g}(q_1)$ can be deduced as well. Informally, if the goals $\mathbf{g}(p_1), \ldots, \mathbf{g}(p_5)$ are satisfied, and if it is the case that when they are satisfied, then $\mathbf{g}(q_1)$ is also satisfied, then $\mathbf{g}(q_1)$

can be considered as satisfied. There is, then, an inference relation from the five goals to $\mathbf{g}(q_1)$ because

$$\{\mathbf{g}(p_1) \wedge \ldots \wedge \mathbf{g}(p_5), \mathbf{k}(\mathbf{g}(p_1) \wedge \ldots \wedge \mathbf{g}(p_5) \rightarrow \mathbf{g}(q_1))\} \mathrel{|\!\sim_{\mathbf{T1}}} \mathbf{g}(q_1) \tag{5.8}$$

and the consequence relation $\mathrel{|\!\sim_{\mathbf{T1}}}$ in **T1** allows the application of *modus ponens*. The conditional $\mathbf{g}(p_1) \wedge \ldots \wedge \mathbf{g}(p_5) \rightarrow \mathbf{g}(q_1)$ is a domain assumption, i.e., $\mathbf{k}(\mathbf{g}(p_1) \wedge \ldots \wedge \mathbf{g}(p_5) \rightarrow \mathbf{g}(q_1))$, because decomposing or refining $\mathbf{g}(q_1)$ onto $\mathbf{g}(p_1), \ldots, \mathbf{g}(p_5)$ requires assuming the conditional that if the former are satisfied, then the latter is satisfied.

The inference relation is used to define derived relations, such as refinement, decomposition, and operationalization. All three are common in languages for RE, and each of them can be defined by adding constraints on the kinds of requirements to allow in an inference relation. E.g., there is a (goal) refinement relation only if there is an inference relation where all premises (except the domain assumption over the implication) are goals, and the conclusion is a goal. When all premises are tasks and the conclusion is a goal, then it is not a refinement, but an operationalization relation. Finally, when all premises are tasks, and the conclusion is a task, then it is a task decomposition relation. The derived relations are revisited later (cf., §5.2).

Conflict is an n-ary relation between members of a minimally inconsistent set of propositions. The identification of available ambulances, i.e., the satisfaction of $\mathbf{g}(p_1)$, can be done at least in two ways. One consists of having every control assistant in the dispatch center manually keep track of available ambulances. Denote this as the task $\mathbf{t}(u_3)$. Another way is to execute two tasks: (i) $\mathbf{t}(u_1)$ which consists of making the list of available ambulances through the dispatch interface, and (ii) $\mathbf{t}(u_2)$ which is to update the list of ambulances when an ambulance is assigned to an incident. The two options are two alternative operationalizations of $\mathbf{g}(p_1)$. Moreover, it is assumed that $\mathbf{t}(u_1)$ and $\mathbf{t}(u_3)$ cannot be satisfied together, i.e., $\mathbf{k}(\mathbf{t}(u_1) \wedge \mathbf{t}(u_2) \rightarrow \bot)$, so that $\bot$ is derived whenever $\mathbf{t}(u_1)$ and $\mathbf{t}(u_2)$ is on the left-hand side of $\mathrel{|\!\sim_{\mathbf{T1}}}$.

Figure 1 gives the refinement of $\mathbf{g}(q_1)$ onto $\mathbf{g}(p_1), \ldots, \mathbf{g}(p_5)$ and the subsequent operationalization of each of these goals onto tasks and domain assumptions.

5.2 Formalization

Semantic Domain. The semantic domain is a set of propositions in natural language, partitioned into the extensions of, respectively the Goal, Domain assumption and Task concepts. Each concept gives one modality over expressions in **T1**, as follows. The symbol for a modality is on the left-hand side below. A proposition is:

- **k**: a domain assumption, if it states a condition that is believed to hold;
- **g**: a goal, if it identifies a desirable condition that does not hold;
- **t**: a task, if it says how to bring about a condition.

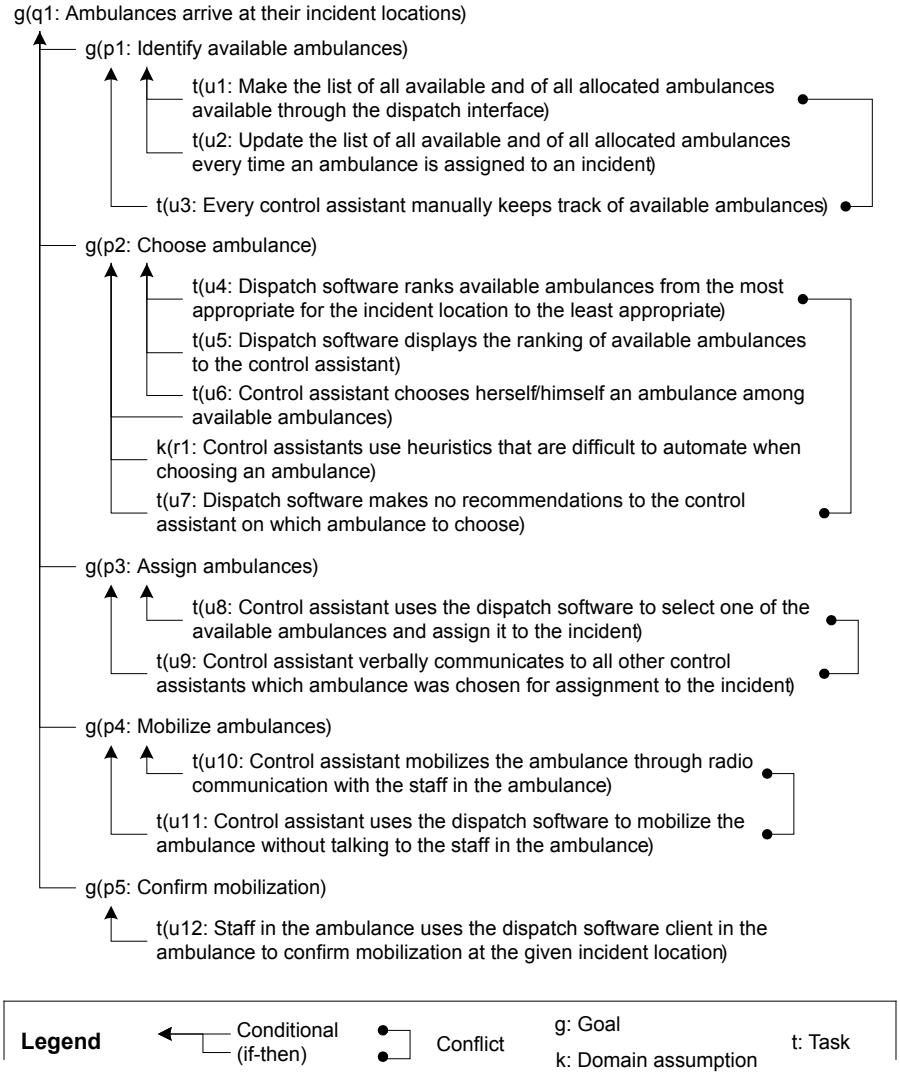

Figure 1: Simple early requirements for LAS specified using **T1**.

There are three primitive relations between propositions: conjunction, defined as usual, implication read as a conditional relation, and consequence, which is nonstandard and defined below.

Syntax. p, q, r, indexed or primed as needed, refer to primitive natural language propositions, i.e., those that need not be further decomposed, because they do not state a relation of **T1** over other propositions. Lowercase and uppercase Greek letters refer to, respectively, expressions and sets of expressions. The

language is the finite set $\mathcal{L}_\mathbf{1}$ of all expressions $\phi \in \mathcal{L}_\mathbf{1}$ which satisfy the following BNF specification:

$$a ::= \mathbf{g}(p) \mid \mathbf{k}(p) \mid \mathbf{t}(p) \tag{5.9}$$

$$b ::= (\bigwedge_{i=1}^{n\geq 1} a_i) \to a \mid (\bigwedge_{i=1}^{n\geq 2} a_i) \to \bot \tag{5.10}$$

$$\phi ::= a \mid \mathbf{k}(b) \tag{5.11}$$

The combinations of the conjunction and implication relations (b above) are always domain assumptions. This reflects the idea that relations between primitive propositions amount to one's assumptions (and not, e.g., desires) about relationships between the conditions stated by the propositions. **T1** does not allow nesting, so that e.g., $\mathbf{g}(\mathbf{t}(p))$ is not an expression.

Semantic Mapping. Propositions and expressions are categorized according to the informal definitions of the concepts of the semantic domain. The semantic mapping function is thereby defined through the informal definitions of the concepts. The domain of the function is $\mathcal{L}_\mathbf{1}$, its codomain the semantic domain.

Consequence Relation. Only the *modus ponens* proof rule can be applied in deduction. All expressions in a requirements database Δ which include the implication connective are considered as axioms in the set $\Delta^{\to}$.

It is assumed throughout the paper the set of axioms $\Delta^{\to}$ is consistent for any Δ. Note that when $\Pi \subseteq \mathcal{L}_\mathbf{1}$, then all axioms apply when deducing from Π and from any other subset of, or set equal to $\mathcal{L}_\mathbf{1}$. To make this explicit, the special subset relation is defined as follows, for any two sets Π and Φ of expressions:

$$\Pi \subseteq_\tau \Phi \stackrel{def}{=} \Pi \subseteq \Phi \text{ and } \Phi^{\to} \subseteq \Pi \tag{$\subseteq_\tau$}$$

$$\Pi \subset_\tau \Phi \stackrel{def}{=} \Pi \subset \Phi \text{ and } \Phi^{\to} \subseteq \Pi \tag{$\subset_\tau$}$$

Definition 5.1. For $\Pi \subseteq_\tau \mathcal{L}_\mathbf{1}$ and $\phi \in \mathcal{L}_\mathbf{1}$, the **consequence relation** $\mid\!\sim_\mathbf{1}$ is such that:

- $\Pi \mid\!\sim_\mathbf{1} \phi$ if $\phi \in \Pi$, or
- $\Pi \mid\!\sim_\mathbf{1} x$ if $\forall 1 \leq n$, $\Pi \mid\!\sim_\mathbf{1} \phi_i$ and $\mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \to x) \in \Pi$.

■

The consequence relation $\mid\!\sim_\mathbf{1}$ is sound with regards to standard entailment $\vdash$ in classical propositional logic, but is incomplete in two ways: it only considers deducing positive atomic facts, and no ordinary proofs based on arguing by contradiction go through, thus being paraconsistent.

5.3 Problem & Solution Concepts

The widely accepted general requirements problem definition is Zave & Jackson's [38] (ZJ hereafter), stating that RE should produce a specification (S) of a design of the system-to-be, which ensures that the system-to-be is consistent with domain assumptions (K) and together with them entails requirements (R): i.e., that $K, S \vdash R$. This definition emphasizes two properties:

1. *Consistency*, in that the operationalization of requirements, i.e., $K \cup S$ must be consistent;
2. *Achievement* of requirements, as $K \cup S$ are only acceptable if they are sufficient to deduce R.

If goals are ZJ's requirements and tasks are their specification, then the requirements problem in **T1** which corresponds to ZJ requirements problem is as follows, whereby the conditions are transferred in the solution concept.

Definition 5.2. Given a set of goals and domain assumptions, find a solution. ■

Both conditions can be evaluated in **T1**. The achievement condition in **T1** equates to asking that all goals are satisfied by tasks and domain assumptions. This in turn requires the introduction of the selection (Select) and operationalization (Op) functions.

The selection function simply returns all expressions sharing a given modality in a set of expressions.

Definition 5.3. The **select function**

$$\mathsf{Select} : \{\mathbf{g}, \mathbf{t}, \mathbf{k}\} \times \wp(\mathcal{L}_\mathbf{1}) \longrightarrow \wp(\mathcal{L}_\mathbf{1}) \tag{5.12}$$

is defined as follows, for $\mathbf{x} \in \{\mathbf{g}, \mathbf{t}, \mathbf{k}\}$ and $\Pi \subseteq \mathcal{L}_\mathbf{1}$:

$$\mathsf{Select}(\mathbf{x}, \Pi) \stackrel{def}{=} \{\phi \mid \phi \text{ has the modality } \mathbf{x} \text{ or } \phi \in \Pi^{\rightarrow}\}. \tag{5.13}$$

The part "or $\phi \in \Pi^{\rightarrow}$" ensures that the output of the select function always includes all axioms. ■

To simplify notation, the following abbreviation is used:

$$\mathsf{Select}(\mathbf{x}, \Pi) \equiv \Pi_\mathbf{x}. \tag{5.14}$$

E.g., $\Pi_\mathbf{g}$ is the set of all goals and axioms in Π.

Given a ϕ, $\mathsf{Op}(\phi)$ should return all sets of tasks and domain assumptions, such that each of these sets is enough to derive ϕ. If a database $\Delta \subseteq \mathcal{L}_\mathbf{1}$ is visualized as an AND/OR graph, then $\mathsf{Op}(\phi)$ returns all sets of leaf nodes connected to ϕ.

Definition 5.4. The **operationalization function**

$$\mathsf{Op} : \Delta \longrightarrow \wp(\wp(\Delta_\mathbf{k} \cup \Delta_\mathbf{t})) \tag{5.15}$$

is defined as follows:

$$\mathsf{Op}(\phi \in \Delta) \stackrel{def}{=} \{\Pi \subseteq_\tau \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}} \mid \Pi \not\vdash_1 \bot, \text{ and } \Pi \vdash_1 \phi, \text{ and } \nexists \Phi \subset \Pi,\ \Phi \vdash_1 \phi\}. \quad (5.16)$$

Every member of $\mathsf{Op}(\phi)$ is a minimal consistent set of tasks and domain assumptions that is sufficient to operationalize ϕ. ■

Informally, $\mathsf{Op}(\phi)$ indicates all ways in which ϕ is operationalized. E.g., $\mathsf{Op}(\mathbf{g}(p_1))$ is the set which includes two sets: (i) $\{\mathbf{t}(u_1), \mathbf{t}(u_2)\}$ and (ii) $\{\mathbf{t}(u_3)\}$.

Definition 5.5. A **solution** to the requirements problem given by a requirements database Δ in **T1** is a set $S \subseteq_\tau \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ of domain assumptions and tasks, which satisfies the following two properties:

1. *Consistency*: $S \not\vdash_1 \bot$;
2. *Achievement*: $\forall \phi \in \Delta_{\mathbf{g}}$ s.t. $\nexists \psi,\ \phi \in \bigcup \mathsf{Op}(\psi)$ and $\exists \Pi \in \mathsf{Op}(\phi)$ s.t. $\Pi \subseteq S$.

■

The Achievement condition is defined in such a way, that the goals to satisfy are all goals which are themselves not refined or operationalized, i.e., which are not children of other requirements in Δ.

For an example of ZJ problem and solution, consider again Figure 1. Assume that all goals there are the set R', all domain assumptions K', and all tasks S'. Given R' and K' initially, the ZJ problem says that one should find tasks $S \subseteq S'$ which satisfy a consistent subset $K \subseteq K'$ of domain assumptions, and a consistent subset of requirements $R \subseteq R'$. The highlighted tasks and domain assumptions in Figure 2 (some of which are shown as arrows going from the tasks to the goals) are one solution to the requirements problem in Figure 1. This can be verified by looking at conflict and conditional relations in Figure 2.

5.4 Derived Relations

The aim now is to start by defining the inference and conflict relations, and consider how they are specialized into oft-used relations in specification languages for RE.

The inference and conflict relations are not primitive in **T1**, as both are defined using conjunction, implication, and consequence relations.

5.4.1 Inference Relation and Its Specialization

Definition 5.6. A requirement $\phi \in \Delta$ stands in the **inference relation** with the requirements $\{\psi_1, \ldots, \psi_n\} \subseteq \Delta$, $n \geq 1$, if and only if:

1. $\mathbf{k}((\bigwedge_{i=1}^n \psi_i) \rightarrow \phi) \in \Delta^{\rightarrow}$;
2. $\nexists \Pi \subseteq_\tau \Delta$ s.t. $\Pi \subset_\tau (\{\psi_1, \ldots, \psi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \vdash_1 \phi$;
3. $\nexists \gamma \in \Delta^{\rightarrow}$ s.t. $\{\gamma\} \cup \{\bigwedge_{i=1}^n \psi_i\} \vdash_1 \bot$.

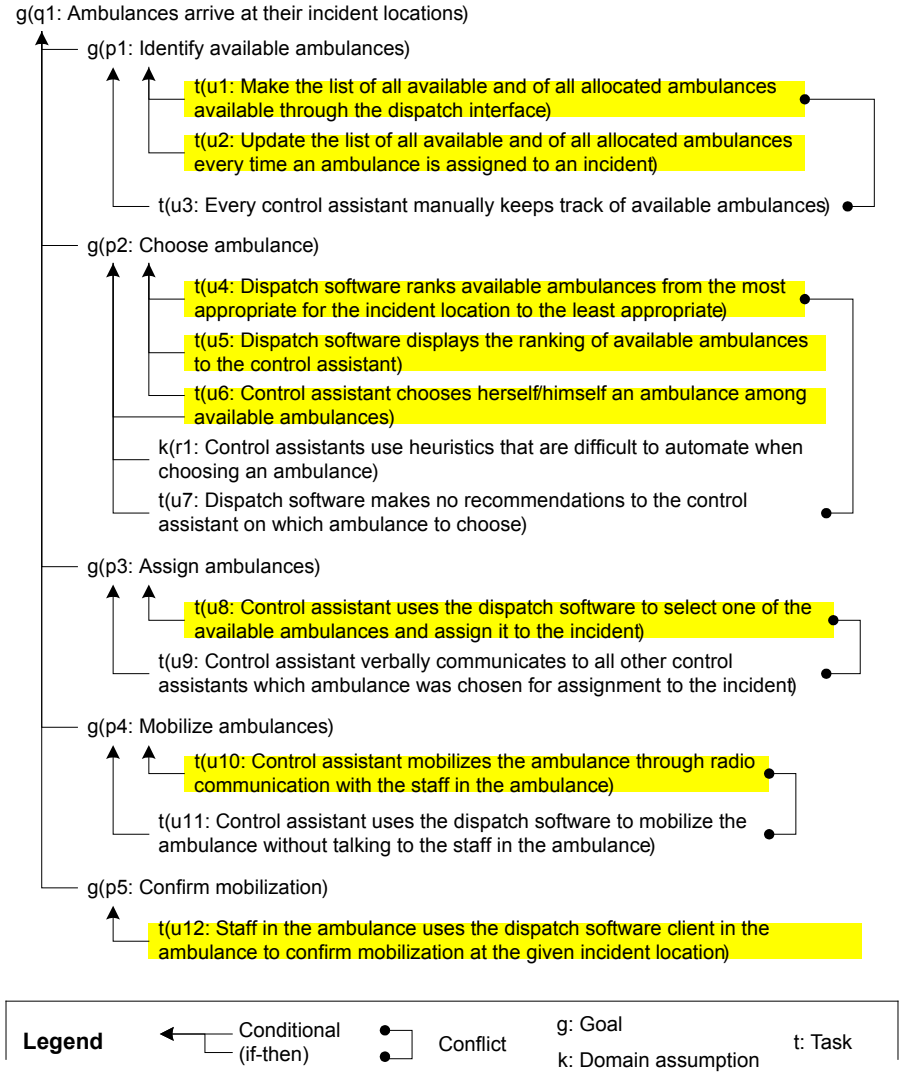

Figure 2: One solution to the requirements problem in Figure 1.

The first condition requires that there be an axiom such that $\{\psi_1, \ldots, \psi_n\} \cup \{\mathbf{k}((\bigwedge_{i=1}^{n} \psi_i) \rightarrow \phi)\} \mathrel{|\!\sim_1} \phi$. The second, minimality condition requires that there be no subset of the premises from which the consequence can be deduced. The third condition requires that the premises be consistent. ■

The inference relation is used to define the concept of *argument* in **T1**. An argument puts together the premises and the conclusion in an inference relation.

Definition 5.7. The pair (Π, ϕ) is an **argument** in Δ if and only if:

1. $\Pi \subseteq_\tau \Delta$,
2. ϕ is in the inference relation to $\Pi \setminus \Delta^{\rightarrow}$.

Two conventions are used: (i) Π is called the set of premises in the argument (Π, ϕ) and ϕ the conclusion of that argument; (ii) the set of all arguments of Δ is:

$$\mathsf{Arg}(\Delta) \stackrel{def}{=} \{(\Pi, \phi) \mid \Pi \subseteq_\tau \Delta\}. \tag{5.17}$$

■

Arguments are used to simplify the definition of the specializations of the inference relation.

The specialization of the inference relation is done by restricting the modalities on the premises and of the conclusion in an inference relation. These constraints and resulting relations are summarized in Table 2. The table shows that four common relations in RE languages can be defined by specializing the inference relation.

Table 2: Inference relation and its specializations, obtained by restricting the modalities on premises and conclusions in an inference relation.

Relation	*Allowed premises*	*Allowed conclusion*
Inference	Any + Axioms	Any
Goal refinement	Goals + Axioms	Goal
Task decomposition	Goals and/or Tasks + Axioms	Task
Goal operationalization	Tasks + Axioms	Goal
Means-ends	Tasks + Axioms	Goal

Goal Refinement. Darimont & van Lamsweerde [10] defined goal refinement as the relationship between a goal being refined and subgoals which refine it, the latter having to satisfy three conditions: (i) be sufficient to deduce the refined goal, (ii) be minimal, and (iii) be consistent. This *goal refinement* relation can be defined as follows in **T1**.

Definition 5.8. A goal $\phi \in \Delta_{\mathbf{g}}$ stands in the **goal refinement** relation with the goals $\{\psi_1, \ldots, \psi_n\} \subseteq \Delta_{\mathbf{g}} \setminus \{\phi\}$, and the former is said to be refined by the latter, if and only if:

1. $\exists \mathbf{k}((\bigwedge_{i=1}^{n} \psi_i) \rightarrow \phi) \in \Delta^{\rightarrow}$;
2. $\nexists \Pi \subseteq_\tau \Delta_{\mathbf{g}}$ s.t. $\Pi \subset_\tau (\{\psi_1, \ldots, \psi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \mathrel{\mid\!\!\sim_{\mathbf{1}}} \phi$;
3. $\nexists \gamma \in \Delta^{\rightarrow}$ s.t. $\{\gamma\} \cup \{\bigwedge_{i=1}^{n} \psi_i\} \mathrel{\mid\!\!\sim_{\mathbf{1}}} \bot$.

The first condition ensures that ϕ can be deduced from the goals and that domain assumption. The second condition is ensures that the set of subgoals is minimal, i.e., that there is no smaller set of goals from which ϕ can be deduced. The third condition requires that there is no domain assumption according to which the subgoals are consistent. ■

Observe that an argument where all premises and the conclusion are goals is an argument where the premises and the conclusion are in the goal refinement relation.

Task Decomposition. Yu & Mylopoulos [37] introduced task decomposition in i-star. It is similar to goal refinement, with two differences: (i) the requirement being refined/decomposed must be a task, and (ii) it can be refined by any combination of goals and tasks.[4] Task decomposition is defined as follows in **T1**.

Definition 5.9. A task $\phi \in \Delta_{\mathbf{t}}$ stands in the **task decomposition** relation with the goals and/or tasks $\{\psi_1, \ldots, \psi_n\} \subseteq \Delta_{\mathbf{g}} \cup \Delta_{\mathbf{t}} \setminus \{\phi\}$, and the former is said to be decomposed onto the latter, if and only if there is an argument (Π, ϕ) of Δ, where $\Pi = \{\psi_1, \ldots, \psi_n\} \cup \Delta^{\rightarrow}$. ■

The task decomposition relation was not formalized in i-star. The definition above assumes that it is reasonable to want to avoid having useless requirements in a decomposition (hence the second – minimality – condition in Definition 5.9) and that the requirements in the decomposition should be consistent (hence the third condition).

Goal Operationalization or Means-Ends. The goal operationalization relation in KAOS [9] is similar to the means-ends relation in i-star. The idea of both is that tasks should be executed in order to satisfy goals. Operationalization in KAOS stands between goals and constraints, whereby a constraint is operational, in the sense that it is formulated in terms of objects and actions available to the agents in/of the system. Means-ends rather emphasizes the role of goals as reasons why tasks are executed, i.e., a task exists in a requirements database because it is a means to a goal. The generic operationalization relation, which captures the idea of both goal operationalization and means-ends is defined in **T1** as follows.

Definition 5.10. A goal $\phi \in \Delta_{\mathbf{g}}$ stands in the **goal operationalization** relation with the tasks $\{\psi_1, \ldots, \psi_n\} \subseteq \Delta_{\mathbf{t}} \setminus \{\phi\}$, and the former is said to be operationalized by the latter, if and only if there is an argument (Π, ϕ) of Δ, where $\Pi = \{\psi_1, \ldots, \psi_n\} \cup \Delta^{\rightarrow}$. ■

5.4.2 Conflict Relation and Its Specialization

Definition 5.11. Requirements $\{\phi_1, \ldots, \phi_n\} \subseteq \Delta$ stand in the **conflict**, for $n \geq 2$, if and only if:

1. $\exists \mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow \bot) \in \Delta^{\rightarrow}$;
2. $\nexists \Pi \subseteq \Delta$ s.t. $\Pi \subset (\{\phi_1, \ldots, \phi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \mathrel{|\!\sim_1} \phi$.

The two conditions state that a conflict relation exists only between members of a minimally inconsistent set of requirements. ■

[4] There is no concept in i-star [37] which corresponds to Domain assumption, so it is not allowed here to have domain assumptions in a decomposition of a task.

The conflict relation is specialized onto *Type-A*, *Type-B*, and *Type-C* conflict relations, the first being conflict relations to tolerate, the second and third being ones to resolve.

Type-A Conflict. Type-A conflicts are used to distinguish between alternative solutions to a requirements problem. Eliminating these conflicts early on may be premature, as this may strongly reduce the number of alternatives being considered. Tolerating these conflicts means leaving them in the requirements database.

Consider Figure 2. All four conflicts in that figure are Type-A, because each of them is between requirements belonging to alternative solutions. Given these four conflicts, there are 2^4 alternative solutions in that figure. Not tolerating these conflicts means eliminating them as the requirements database gets constructed: in Figure 2, suppose that $\mathbf{g}(q_1)$ was identified first, and then $\mathbf{g}(p_1)$. Not tolerating the conflict between $\mathbf{t}(u_1)$ and $\mathbf{t}(u_3)$ would consist of choosing to keep one of these two, *regardless of other requirements*. This would eliminate the conflict between $\mathbf{t}(u_1)$ and $\mathbf{t}(u_3)$, but it would also reduce the number of alternative solutions from 2^4 to 2^3.

To define the Type-A conflict relation, the concept of *alternative* is introduced.

Definition 5.12. Given a set $\Pi \subseteq_\tau \Delta$, $\Phi^\Pi \subset_\tau \Pi$ is an **alternative** in Π if and only if:

1. all members of Π are in conflict;
2. $\Phi^\Pi \not\vdash_1 \bot$;
3. $\forall \Psi \subseteq_\tau \Pi$ s.t. $\Psi \not\vdash_1 \bot$, $\Phi^\Pi \not\subset_\tau \Psi$;
4. $\Phi_\Pi \not\subseteq_\tau \Pi_{\mathbf{k}}$.

An **alternative** is a set of requirements which (1) is a subset of a conflicting set of requirements (i.e., Φ^Π is an alternative in Π, and Π must be in conflict), (2) is consistent, (3) is a maximally consistent subset of Π, and (4) cannot include only domain assumptions.

The set of all alternatives in Π is denoted $\mathsf{Alt}(\Pi)$. ■

An alternative identifies one potential resolution of a Type-A conflict. Informally, one can resolve the Type-A conflict by choosing an alternative over others in that conflict.

The important property of an alternative is that it *cannot only include* domain assumptions. An alternative must include a goal or task because the purpose of an alternative is precisely to allow the distinction between candidate solutions. A solution will be made by the consistent combination of alternatives, which also together satisfy the properties of a solution given earlier.

Definition 5.13. There is a **Type-A conflict** relation between the members of Π if and only if:

1. all members of Π are in a conflict relation;

2. $|\mathsf{Alt}(\Pi)| \geq 2$.

As soon as there are two alternatives in a conflict, it is a Type-A conflict. ■

In Figure 2, every conflict is a Type-A conflict. As an alternative contains goal and/or tasks, and a Type-A conflict involves at least two alternatives, it is useful to tolerate Type-A conflicts. Their premature resolution results in the elimination of candidate solutions.

Type-B Conflict. When there is at most one alternative in a conflict, it is a Type-B conflict. Informally, a Type-B conflict involves domain assumptions which are blocking the satisfaction of goals or the execution of tasks.

Definition 5.14. There is a **Type-B conflict** relation between the members of Π if and only if:

1. all members of Π are in a conflict relation;
2. $|\mathsf{Alt}(\Pi)| = 1$.

The set of blockers is the set $\mathsf{Block}(\Pi) \stackrel{def}{=} \Pi \setminus \mathsf{Alt}(\Pi)$. ■

There are no Type-B conflicts in Figure 2. For illustration, suppose that $\mathbf{g}(p_6$: Incident location identified automatically) and that $\mathbf{k}(r_2$: Callers report imprecise incident location), and that there is an axiom $\mathbf{k}(\mathbf{g}(p_6) \wedge \mathbf{k}(r_2) \rightarrow \bot)$. If there was no domain assumption $\mathbf{k}(r_2)$, one could choose to find a way to operationalize $\mathbf{g}(p_6)$ and include it in a candidate solution. $\mathbf{k}(r_2)$ itself is not desirable, it is simply believed to hold; since $\mathbf{g}(p_6)$ is desired, this conflict can be informally interpreted as that $\mathbf{k}(r_2)$ is blocking $\mathbf{g}(p_6)$.

The purpose of the Type-B relation is to make explicit cases when assumptions about the domain are blocking goals and/or tasks. A Type-B thus seems to suggest how to resolve the conflict, namely, by finding good reasons either to keep the blockers or to eliminate them.

Type-C Conflict. A conflict relation where there are no alternatives is a Type-C conflict. It involves a minimally inconsistent set which includes only domain assumptions.

Definition 5.15. There is a **Type-C conflict** relation between the members of Π if and only if:

1. all members of Π are in a conflict relation;
2. $|\mathsf{Alt}(\Pi)| = 0$.

There are no blockers in a Type-C conflict, as all requirements involved in the conflict are domain assumptions. ■

Conflict, Obstruction, Divergence. In KAOS, the Conflict relation is also a minimally inconsistent set of requirements. The main difference is in the consequence relation, which is classical there, and paraconsistent here. Obstruction and Divergence are two relations, also in KAOS, which involve domain assumptions that block, in the sense discussed above, the satisfaction of a goal or the execution of a task.

Table 3 summarizes the translation of conflict relations identified in Robinson, Pawlowski & Volkov's survey [26] into **T1**. Conflicts listed in that table cannot obtain in **T1** definitions as convenient as, e.g., Type-A, Type-B, or Type-C relations. In a propositional formalism, there is no elegant way to formally talk about instances of classes, and their deviations: a proposition stating the deviation of an instance will be different from a proposition stating normal behavior of other instances, but there is no relation which would say that the two propositions talk about instances of the same class.

5.5 Database Interface

The database interface should include operations that return methodologically relevant answers from a given requirements database Δ, which includes the axioms, domain assumptions that are not axioms, tasks, and goals, as well as the various kinds of inference and conflict relations between them.

The methodological questions are essentially those that support the incremental construction of solutions. Asking any such question should result in information that helps when choosing what to add, remove, or revise in a given requirements database towards the construction of solutions.

To characterize operations that are relevant to this aim, filters can be defined, whereby an operation should be an implementation of a filter. A filter is simply an intensionally defined set. Calling it a filter applies in that the set will include only information having certain properties, and these properties will have some relevance in the construction of solutions, in the identification of solutions, and the identification and resolution of conflicts. The method applied to define these operations involves choosing a question to answer, then determining the necessary properties of expressions (their modality, their participation in inference and conflict relations) to find in the answer.

The result is an operation which returns the answers to a question of interest. This method is only relevant for filters which should return implicit information. Relevant operations on explicit information are obvious: an operator per modality, which returns all expressions of a given modality (i.e., an operator which returns, for a given Π, $\Pi_{\mathbf{x}}$) and an operator which returns all axioms.

For a requirements database Δ, sets of answers of interest are defined below. The definition of specific consequence relations comes after the answer sets are defined.

Proto-Solutions. A proto-solution is a maximally consistent subset of Δ. It is a *proto*-solution because, by being maximally consistent, it *can* be made into a solution. E.g., it may lack operationalizations for some goal, so adding an

Table 3: Translation to **T1** of the conflict relation types in Robinson, Pawlowski & Volkov's survey [26].

Relation in the survey	*Corresponding relation in* **T1**
Process-level deviation: deviation of the actual process of developing the system from the predefined process.	Either a Type-B conflict, in which the blocker is a domain assumption stating the deviation between the planned and actual development process, or a Type-C conflict where one of the domain assumptions states that deviation.
Instance-level deviation: an instance of an implemented class violates a requirement.	A Type-B conflict, which has one blocking domain assumption. That domain assumption names the instance responsible for the violation, and the alternative in the Type-B is the requirement violated by that instance.
Terminology clash (also *Structure clash*): a member of the semantic domain is being referred to using more than one symbol/expression.	None of the conflict relations captures terminology clashes. The terminology clash is an error in the use of the formalism.
Designation clash: a symbol/expression refers to two or more different members of the semantic domain.	As for the terminology clash, a designation clash is an error in the use of the formalism, and cannot be captured in the formalism.
Conflict: a set of requirements is logically inconsistent.	Conflict relation.
Divergence (also *Obstruction*): A set of requirements is logically inconsistent when a certain sequence of events can occur.	A Type-B conflict, where the blocked requirements are an alternative and the blocker is a domain assumption describing the problematic sequence of events.
Competition: A kind of divergence where particular instances of a requirement can cause a divergence.	A Type-B conflict, which has one blocking domain assumption. That domain assumption names the instance responsible for the violation, and the alternative in the Type-B is the requirement violated by that instance.

operationalization to a proto-solution is relevant. It may also be missing a goal, because the goal is involved in a conflict; in such a case, the maximally consistent subset still remains a relevant starting point for the construction of a solution.

The set of all proto-solutions is $\mathsf{AMCon}(\Delta)$, and to find it, it is necessary to know the set of all consistent subsets of Δ, $\mathsf{ACon}(\Delta)$:

$$\mathsf{ACon}(\Delta) = \{\Pi \subseteq_\tau \Delta \mid \Pi \not\vdash_1 \bot\}, \tag{5.18}$$

$$\mathsf{AMaxCon}(\Delta) = \{\Pi \in \mathsf{ACon}(\Delta) \mid \forall \Phi \in \mathsf{ACon}(\Delta),\ \Pi \not\subset_\tau \Phi\}. \tag{5.19}$$

$\mathsf{AMaxCon}(\Delta)$ and $\mathsf{ACon}(\Delta)$ are independent of modalities. They are consequently not specific to databases of requirements: they remain relevant (as long as the consequence relation is relevant) whenever there is a need to find plausible parts of an inconsistent database. It is by restricting answer sets using modalities that they become specific to requirements databases.

Goals. Given a proto-solution, one needs to decide how to change it towards the status of a solution. Let $\Pi \in \mathsf{AMaxCon}(\Delta)$ be the proto-solution.

To decide how to change Π, start by identifying top-level goals in Δ:

$$\mathsf{AGTop}(\Delta) = \{\phi \mid \phi \in \Delta_{\mathbf{g}} \text{ and } \nexists \psi \in \Delta,\ \phi \in \bigcup \mathsf{Op}(\psi)\} \tag{5.20}$$

If $\mathsf{AGTop}(\Delta) \not\subseteq \Pi$, then there are top-level goals which are involved in conflicts. No proto-solution will include them. No proto-solution can consequently be made into a solution before conflicts involving top-level goals are resolved.

Suppose that $\mathsf{AGTop}(\Delta) \subseteq \Pi$, so that all top-level goals are in the proto-solution. It is then relevant to determine if some of these goals lack operationalizations in Π. The set

$$\mathsf{AGnOp}(\Pi) = \{\phi \mid \phi \in \Pi_{\mathbf{g}} \text{ and } |\mathsf{Op}(\phi)| \geq \mathsf{n}\} \tag{5.21}$$

includes all goals which have at least n operationalizations in Π. If $\mathsf{n} = 1$ and $\mathsf{AGTop}(\Delta) \subseteq \mathsf{AG1Op}(\Pi)$, then all top-level goals are operationalized in Π, so that Π is a solution according to Definition 5.5.

If $\mathsf{n} = 1$ and $\mathsf{AGTop}(\Delta) \not\subseteq \mathsf{AG1Op}(\Pi)$, then operationalizations of goals in $\mathsf{AGTop}(\Delta) \setminus \mathsf{AG1Op}(\Pi)$ should be made so as to be consistent with Π and added to Δ.

Tasks. If a task does not participate in a premise of an argument, then its role in the database should be reconsidered. The following set returns all such tasks:

$$\mathsf{ATNoArgPrem}(\Delta) = \{\phi \in \Delta_{\mathbf{t}} \mid \nexists (\Pi, \psi) \in \mathsf{Arg}(\Delta),\ \phi \in \Pi\}. \tag{5.22}$$

When a task is in $\mathsf{ATNoArgPrem}(\Delta)$ it may be the case that there is an axioms which includes that task in its antecedent, but not all of the antecedents are present in the database. E.g., there is $\phi \wedge \psi \to \gamma$ in Δ, but $\phi \in \Delta$ and $\psi \notin \Delta$.

Detecting this requires an operation which returns, for a given requirement, all axioms where that requirement is in the antecedent and the consequent is not $\perp$. Such an operation involves no deductions on Δ. Let this operation be as follows:

$$\mathsf{AxiomAnte}(\phi) = \{\psi \in \Delta^{\rightarrow} \mid \phi \text{ is among the antecedents of } \psi \text{ and } \perp \text{ is not the consequent of } \psi\}. \tag{5.23}$$

The set of "aimless" tasks is then:

$$\mathsf{ATAimless}(\Delta) = \mathsf{ATNoArgPrem}(\Delta) \setminus \{\phi \mid \mathsf{AxiomAnte}(\phi) = \emptyset\}, \tag{5.24}$$

which are all tasks that do not participate in arguments, and there are no axioms in which such tasks are among the antecedents. It is useful to reconsider the role of these tasks in the database.

A subset of $\mathsf{ATNoArgPrem}$ that of interest for the construction of solutions is the subset of tasks which participate in no operationalizations of goals. This set is as follows:

$$\mathsf{ATNoOp}(\Delta) = \{\phi \mid \phi \in \Delta_{\mathbf{t}} \text{ and } \nexists \psi \in \Delta_{\mathbf{g}},\ \phi \in \bigcup \mathsf{Op}(\psi)\}. \tag{5.25}$$

It is straightforward to show that $\mathsf{ATNoOp}(\Delta) \subseteq \mathsf{ATNoArgPrem}(\Delta)$, and that $\mathsf{ATNoArgPrem}(\Delta) \not\subset \mathsf{ATNoOp}(\Delta)$.

The level of detail of lowest-level tasks can be compared. Lowest-level tasks are those which cannot be deduced from other information in Δ. If the level of detail of lowest-level tasks differs strongly, it may be relevant to decompose tasks which are judged to lack detail. This in turn can lead to the detection of conflicts. The set $\mathsf{ATNoArgConc}(\Delta)$ includes all tasks which are not conclusions of arguments:

$$\mathsf{ATNoArgConcl}(\Delta) = \{\phi \in \Delta_{\mathbf{t}} \mid \nexists(\Pi, \phi) \in \mathsf{Arg}(\Delta)\}. \tag{5.26}$$

As for $\mathsf{ATNoArgPrem}$, note that there may be axioms which have a task ϕ as their consequent, but arguments for ϕ may still be missing because some of the antecedents may be missing from Δ. Hence the following operation:

$$\mathsf{AxiomCons}(\phi) = \{\psi \in \Delta^{\rightarrow} \mid \phi \text{ is the consequent of } \psi\}, \tag{5.27}$$

which gives all axioms with ϕ as their consequent. The following set:

$$\mathsf{ATBasic}(\Delta) = \mathsf{ATNoArgConcl}(\Delta) \setminus \{\phi \mid \mathsf{AxiomCons}(\phi) = \emptyset\}, \tag{5.28}$$

includes all "basic" tasks, which are neither conclusions of arguments, nor are consequents in axioms. If a task is in this set, then no attempts have been made to decompose or refine it.

Domain assumptions. For domain assumptions which are *not* axioms, the interesting questions are analogous to those for tasks. Hence the following answer sets:

$$\mathsf{AKNoArgPrem}(\Delta) = \{\phi \in \Delta_{\mathbf{k}} \mid \nexists(\Pi, \psi) \in \mathsf{Arg}(\Delta),\ \phi \in \Pi\}, \tag{5.29}$$

$$\mathsf{AKAimless}(\Delta) = \mathsf{AKNoArgPrem}(\Delta) \setminus \{\phi \mid \mathsf{AxiomsAnte}(\phi) = \emptyset\}, \tag{5.30}$$

$$\mathsf{AKNoOp}(\Delta) = \{\phi \mid \phi \in \Delta_{\mathbf{k}} \text{ and } \nexists\psi \in \Delta_{\mathbf{g}},\ \phi \in \bigcup \mathsf{Op}(\psi)\}, \tag{5.31}$$

$$\mathsf{AKNoArgConcl}(\Delta) = \{\phi \in \Delta_{\mathbf{t}} \mid \nexists(\Pi, \phi) \in \mathsf{Arg}(\Delta)\}, \tag{5.32}$$

$$\mathsf{AKBasic}(\Delta) = \mathsf{AKNoArgConcl}(\Delta) \setminus \{\phi \mid \mathsf{AxiomsCons}(\phi) = \emptyset\}. \tag{5.33}$$

Uncommon Relations. The operationalization function induces the operationalization relation, which states that a requirement is operationalized by combinations of tasks and/or domain assumptions (cf., Definition 5.4). This, together with the other relations derived from the inference relation, is not enough to account for all potential combinations of modalities in an inference relation. There is no explicit rule in **T1** which forbids that, say, goals be the premises and a domain assumption be a conclusion in an inference relation.

Table 4: Additional combinations of modalities in an inference relation. These relations are additional specializations of the inference relation.

Relation	*Allowed premises*	*Allowed conclusion*
Inference	Any + Axioms	Any
Assumption inference	Any + Axioms	Domain assumption
Task inference	Any + Axioms	Task
Goal inference	Any + Axioms	Goal

Table 4 gives a clearer specialization of the inference relation than Table 2. In the former, there are no constraints on modalities in premises, but on the conclusion, and the three inference variants given cover all three modalities in **T1**. The definitions of the three specializations of inference are not given, as they are straightforward adaptations of Definition 5.6. It is also not difficult to see that some of the relations in Table 2 are specializations of those in Table 4.

Uncommon relations are those where the combination of the modalities in the premises and conclusion are unexpected, given the usual relations used in RE formalisms, such as goal refinement, task decomposition, and so on (cf., §5.4). Examples include a domain assumption being decomposed onto goals, tasks decomposed onto goals, and so on. It is useful to consider such relations individually to see whether they make sense, or signal errors in the use of the formalism. The following answer sets are useful to this aim:

$$\mathsf{AGUnc}(\Delta) = \{\phi \in \Delta_{\mathbf{g}} \mid \exists(\Pi, \psi) \in \mathsf{Arg}(\Delta) \text{ and } \psi \in \Delta_{\mathbf{t}} \cup \Delta_{\mathbf{k}} \text{ and } \phi \in \Pi\}, \tag{5.34}$$

$$\mathsf{ATUnc}(\Delta) = \{\phi \in \Delta_{\mathbf{t}} \mid \exists(\Pi, \psi) \in \mathsf{Arg}(\Delta) \text{ and } \psi \in \Delta_{\mathbf{k}} \text{ and } \phi \in \Pi\}. \tag{5.35}$$

AUncG returns all goals which participate in premises of inference relations that conclude a task or a domain assumption. Informally, the goal is in a refinement of a task/domain assumption. AUncT returns the tasks which are in premises of inferences that conclude domain assumptions, i.e., tasks which refine a domain assumption.

Conflicts. The following answer sets serve to understand better the conflict relations in a requirements database:

$$\mathsf{AC}(\Delta) = \{\Pi \mid \Pi \mathrel{\mid\!\sim_1} \bot \text{ and } \nexists \Phi \subset \Pi,\ \Phi \mathrel{\mid\!\sim_1} \bot\}, \tag{5.36}$$

$$\mathsf{ACTypeA}(\Delta) = \{\Pi \in \mathsf{AC}(\Delta) \mid |\mathsf{Alt}(\Pi)| \geq 2\}, \tag{5.37}$$

$$\mathsf{ACTypeB}(\Delta) = \{\Pi \in \mathsf{AC}(\Delta) \mid |\mathsf{Alt}(\Pi)| = 1\}, \tag{5.38}$$

$$\mathsf{ACTypeC}(\Delta) = \{\Pi \in \mathsf{AC}(\Delta) \mid |\mathsf{Alt}(\Pi)| = 0\}, \tag{5.39}$$

$$\mathsf{ABlockers}(\Delta) = \{\phi \in \Pi \mid \Pi \in \mathsf{ACTypeB}(\Delta) \text{ and } \phi \notin \mathsf{Alt}(\Pi) \text{ and } \phi \notin \Delta^{\rightarrow}\}, \tag{5.40}$$

AC is the set of all minimally inconsistent subsets of Δ. Requirements in each of these subsets is in conflict. ACTypeA, ACTypeB, and ACTypeC return subsets of Δ which are in one of the three specializations of the conflict relation. ABlockers is the set of all requirements which act as blockers in Type-B conflicts.

5.6 Discussion

T1 has a simple ontology which has three primitive concepts (**g**, **k**, **t**) and three primitive relations (conjunction, implication, and consequence). It was shown that already this set of notions allows the definition of a wide variety of relations common in RE formalisms. **T1** can thereby capture various kinds of refinement and decomposition, and allows a definition of the requirements problem and solution concepts which correspond to ZJ's definitions. Limitations of **T1** will become clear as additional Techne formalisms are introduced.

6 T2

T2 extends **T1** with *agency modalities*, which capture four kinds of primitive relations:

1. *Responsibility*: that a role is responsible for the achievement of a goal, the maintenance of a domain assumption, or the execution of a task;

2. *Ability*: that an agent is able to achieve a goal, maintain a domain assumption, or execute a task;

3. *Occupancy*: that an agent should occupy a role, and thereby should act to discharge the responsibilities of the role;

4. *Commitment*: that an agent chooses to achieve a goal, maintain a domain assumption, or execute a task.

All four relations are binary, and the syntax of **T2** is such that responsibility, ability, and commitment are each between roles and requirements (i.e., goals, tasks, and domain assumptions which are not axioms), and occupancy and commitment are both from agents to roles.

The four relations require the introduction of agents and roles as extensions of the corresponding Agent and Role primitive concepts. Agency modalities are combinations of an agent or of a role, and of a relation of that agent/role to a requirement. E.g., $\mathsf{R}_i\mathbf{g}(p)$ says that role R_i is responsible for the satisfaction of the goal $\mathbf{g}(p)$.

Responsibility and occupancy can be viewed as stating the desired combinations of responsibilities and of agents discharging these responsibilities. Responsibility bundles requirements by associating them to a role, independently of whether one can actually find an agent capable of satisfying all the bundled requirements. Responsibility is a desired way of grouping requirements in order to distributed them across agents. Occupancy is used to state desired allocations of agents to roles. In this same perspective, ability and commitment are, respectively, about which agents are able to discharge responsibilities and which of them choose (i.e., commit) to do so.

Overall, **T1** was blind to agents, roles, and the relations relevant for the distribution and discharging of responsibilities to satisfy requirements, while **T2** makes these considerations explicit.

6.1 Illustration

The original manual LAS involved the following major activities [8]:

> *Call taking:* Control Assistants at the LAS control center would write down the details on a pre-printed form. The assistants would then locate the incident co-ordinates in their map book and place the completed forms on a conveyor belt transporting all the forms to a central collection point.
>
> *Resource identification:* Another assistant would collect the forms, scan the details, identify potential calls, and allocate them to one of the four regional resource allocators. The appropriate resource allocator would examine the incident forms, consult ambulance status and location information provided by the radio operator, consult the remaining forms maintained in the allocation box for each vehicle,

and finally decide on which resource (ambulance) to mobilize. The ambulance details would be entered on the form.

Resource mobilization: The forms would be passed to a dispatcher who would then phone the relevant ambulance station (if that is where the ambulance was assumed to be) or pass the mobilization instructions to the radio operator, if the ambulance was known to be mobile.

In **T1**, the fragment "Control Assistants at the LAS control center would write down the details on a pre-printed form" could be written as a goal, operationalized via a goal and task:

$\mathbf{g}(p_1$: Control Assistant acquires incident details);

$\mathbf{t}(p_2$: Control Assistant writes incident details on a pre-printed form);

whereby the task would operationalize the goal via the domain assumption $\mathbf{k}(\mathbf{t}(p_2) \rightarrow \mathbf{g}(p_1))$.

In **T2**, this same fragment is rewritten as follows. Firstly, the roles and agents are taken out of propositions, which gives:

$\mathbf{g}(p_1$: Acquire incident details);

$\mathbf{t}(p_2$: Write incident details on a pre-printed form).

Apart from eliminating roles and agents from propositions, other departures from **T1** are as follows:

- Control Assistant is a role, denoted R_1, in LAS. This role is responsible for the execution of the task above, which is written $\mathsf{R}_1\mathbf{t}(p_2)$.
- If $\mathbf{k}(\mathbf{t}(p_2) \rightarrow \mathbf{g}(p_1)) \in \Delta$, then the task operationalizes the goal. If the goal is the responsibility of R_1 and the goal is the responsibility of R_2, then the three-place derived relation between the two roles and the domain assumption $\mathbf{k}(\mathbf{t}(p_2) \rightarrow \mathbf{g}(p_1))$ can be read as that R_2 delegates the satisfaction of the goal $\mathbf{g}(p_1)$ to the role R_1.
- The domain assumption remains the same, $\mathbf{k}(\mathbf{t}(p_2) \rightarrow \mathbf{g}(p_1))$. *Contra* **T1**, it is now not enough to have this domain assumption to operationalize $\mathbf{g}(p_1)$: it is also necessary to identify an agent able to execute the task, and commits to execute the task. In the case of LAS, there may be agents, e.g., A (employees in the manual system) who is able to execute the task, written $\mathsf{A}\mathbf{t}(p_2)$, and who also commit to the task, written $\mathsf{cA}\mathbf{t}(p_2)$.
- Operationalization becomes more complicated in **T2**, as it requires additional conditions to be satisfied compared to operationalization in **T1**. To operationalize $\mathbf{g}(p_1)$, there should be, as in **T1**, the domain assumption $\mathbf{k}(\mathbf{t}(p_2) \rightarrow \mathbf{g}(p_1))$, but also (i) $\mathsf{R}\mathbf{t}(p_2)$, i.e., that the task is the responsibility of a role, (ii) $\mathsf{A}\mathbf{t}(p_2)$, that there is an agent able to execute the task, (iii) $\mathsf{cA}\mathbf{t}(p_2)$, that the agent commits to the task, and finally (iv) $\mathsf{O}(\mathsf{R}, \mathsf{A})$, that the agent occupies the role responsible for the task.

To illustrate this last point, consider the Resource mobilization fragment quoted above. A general goal can be set, $\mathbf{g}(p_3)$, where p_3 is for *Pass mobilization instructions to ambulance*. Suppose that this goal was made the responsibility of some role R_4, so that $\mathsf{R}_4\mathbf{g}(p_3)$. This is a problem, since the agent occupying this role would have to radio the instructions to ambulances outside stations, and phone ambulances in stations. The simpler option is to refine $\mathbf{g}(p_3)$ onto $\mathbf{g}(p_4)$ and $\mathbf{g}(p_5)$, where p_4 is for *Pass mobilization instructions to ambulances in stations* and p_5 is for *Pass mobilization instructions to ambulances outside stations*. To complete this refinement, the domain assumption $\mathbf{k}(\mathbf{g}(p_4) \wedge \mathbf{g}(p_5) \rightarrow \mathbf{g}(p_3))$ is needed, as well as two roles, R_5 and R_6, such that $\mathsf{R}_5\mathbf{g}(p_4)$ and $\mathsf{R}_6\mathbf{g}(p_5)$. R_4 can be viewed as an intermediary role: the responsibilities of this role have been refined, and new roles defined for these more detailed responsibilities, there is no need in a solution to keep the intermediary role. Agents will be discharging the responsibilities of that role by occupying other roles.

A final, important departure of **T2** from **T1** is the change in the conceptualization of the solution concept. In **T1**, it was necessary to find domain assumptions and tasks, and goals were not allowed. Suppose now that there is an agent A such that $\mathsf{A}\mathbf{g}(p)$. Assume further that $\mathsf{R}\mathbf{g}(p)$ and $\mathsf{O}(\mathsf{R}, \mathsf{A})$. In words, A occupies the role R, R is responsible for $\mathbf{g}(p)$, and A is able to achieve p, i.e., to satisfy the goal $\mathbf{g}(p)$. In this case, it is reasonable to decide that one knows enough, and that $\mathbf{g}(p)$ is already operationalized by being assigned to an agent capable of satisfying the goal.

6.2 Formalization

Semantic Domain. Objects in the semantic domain are natural language propositions, agents, and roles. Propositions in natural language are partitioned into the extensions of Goal, Domain assumption, and Task concepts. Instances of goals, domain assumptions, and tasks are related to agents and roles via three relations. A combination of agent/role and of a relation to a proposition defines a modality on the proposition, as follows:

$\mathsf{R}_i\mathbf{g}(p)$: role referred to by the symbol R_i is responsible to achieve p;

$\mathsf{R}_i\mathbf{k}(p)$: R_i is responsible for the maintenance of p;

$\mathsf{R}_i\mathbf{t}(p)$: R_i is responsible for the execution of p;

$\mathsf{A}_i\mathbf{g}(p)$: agent referred to by the symbol A_i is able to achieve p;

$\mathsf{A}_i\mathbf{k}(p)$: A_i is able to maintain p;

$\mathsf{A}_i\mathbf{t}(p)$: A_i is able to execute p;

$\mathsf{cA}_i\mathbf{g}(p)$: A_i commits to achieve p;

$\mathsf{cA}_i\mathbf{k}(p)$: A_i commits to maintain p;

$\mathsf{cA}_i\mathbf{t}(p)$: A_i commits to execute p.

There are five primitive relations. *Conjunction* and *implication* are as in **T1**. *Consequence* is again nonstandard, and is defined below. *Responsibility*, *ability* and *commitment* are informally interpreted as above. Finally, *occupancy* ($\mathsf{O}(\mathsf{R},\mathsf{A})$) relates a role to an agent who occupies that role, i.e., the latter is expected to discharge the responsibilities of the former.

Syntax. p, q, r, indexed or primed as needed, refer to natural language propositions. A and R are indexed, and denote, respectively, an agent and a role. The language is a finite set $\mathcal{L}_\mathbf{2}$ of all expressions $\phi \in \mathcal{L}_\mathbf{2}$ which satisfy the following BNF specification:

$$a ::= \mathbf{g}(p) \mid \mathbf{k}(p) \mid \mathbf{t}(p) \tag{6.41}$$

$$b ::= \mathsf{R}a \mid \mathsf{A}a \mid \mathsf{cA}a \tag{6.42}$$

$$c ::= a \mid b \mid \mathsf{O}(\mathsf{R},\mathsf{A}) \tag{6.43}$$

$$d ::= (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow c \mid (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow \bot \tag{6.44}$$

$$\phi ::= c \mid \mathbf{k}(d) \tag{6.45}$$

Conjunction and implication follow the grammar of **T1**. Syntax of **T2** symbols for agents, roles, and the responsibility, ability and occupy relations. Below are illustrations of the uses and limits of the grammar:

- Responsibility is only for roles, ability for agents. Agents are given responsibilities only by occupying roles, to reflect the idea that roles are placeholders for agents within the system-to-be.
- To keep the combinations of modalities simple in **T2**, agency modalities are not allowed closer to propositions than core modalities: e.g., $\mathbf{g}(\mathsf{R}a)$ is not an expression. An expression stating responsibility, ability, or commitment also cannot be a goal or a task. It can be a domain assumption, but only when that domain assumption is an axiom.
- Responsibility, ability, commitment, or occupancy relations can be the conclusion in an inference relation: e.g., $\mathbf{k}(p) \rightarrow \mathsf{R}\mathbf{g}(q)$ says that if the goal $\mathbf{g}(p)$ is satisfied, then R is responsible for the satisfaction of $\mathbf{g}(q)$. E.g., in the Resource identification description for the manual LAS, an assistant, denote it R_1, collects the forms on incidents and allocates them to a resource allocator, R_2. If it is the case that the agent A occupying R_1 will occupy the role R_2 also when $\mathbf{k}(p :$ Dispatch center overloaded), then this can be written: $\mathbf{k}(\mathbf{k}(p) \wedge \mathsf{O}(\mathsf{R}_1,\mathsf{A}) \rightarrow \mathsf{O}(\mathsf{R}_2,\mathsf{A}))$.
- Responsibility, ability, commitment, and occupancy relations can be in conflict, and they can individually be made impossible. E.g., if an agent A cannot occupy two roles, R_1 and R_2, then $\mathbf{k}(\mathsf{O}(\mathsf{R}_1,\mathsf{A}) \wedge \mathsf{O}(\mathsf{R}_2,\mathsf{A}) \rightarrow \bot)$. If it is known that A is not able to ϕ, then $\mathsf{A}\phi \rightarrow \bot \in \Delta$.

Semantic Mapping. The semantic mapping function relates members of $\mathcal{L}_{\mathbf{2}}$ to objects in the semantic domain. The informal discussions given above explain which symbols in $\mathcal{L}_{\mathbf{2}}$ map to which objects and relations in the semantic domain.

Consequence Relation. The consequence relation $\mid\!\sim_{\mathbf{2}}$ is defined in the same way as $\mid\!\sim_{\mathbf{1}}$, even though the formal languages for each of these relations are different.

Definition 6.1. For $\Pi \subseteq_{\tau} \mathcal{L}_{\mathbf{2}}$ and $\phi \in \mathcal{L}_{\mathbf{2}}$, the **consequence relation** $\mid\!\sim_{\mathbf{2}}$ is such that:

- $\Pi \mid\!\sim_{\mathbf{2}} \phi$ if $\phi \in \Pi$, or
- $\Pi \mid\!\sim_{\mathbf{2}} x$ if $\forall 1 \leq n$, $\Pi \mid\!\sim_{\mathbf{2}} \phi_i$ and $\mathsf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow x) \in \Pi$.

■

Remarks on soundness and completeness made for $\mid\!\sim_{\mathbf{1}}$ apply for $\mid\!\sim_{\mathbf{2}}$.

6.3 Problem & Solution Concepts

The requirements problem in **T2** relies on an extended notion of operationalization compared to **T1**. A requirement can now be operationalized in two ways:

- by assigning that goal to a role, and then having an agent occupy that role, whereby the the agent is able to satisfy that goal and commits to do so; it is not necessary in this case to know what tasks that agent is going to execute to satisfy the goal;
- by identifying tasks and domain assumptions from which the goal can be deduced (i.e., by searching for an oprationalization of the goal in the sense of **T1**), assigning these tasks to roles, and having capable agents commit to the tasks in these roles.

The change to the requirements problem is straightforward.

Definition 6.2. Given a set of goals, domain assumptions, and agents, find a solution. ■

This formulation is intentionally a very general one. It makes no assumptions on the knowledge of roles, tasks, responsibility, ability, commitment, and occupancy. It is instead assumed that there are agents, goals and conditions already holding in the environment, and the problem consists of finding a solution to how agents will satisfy the goals by occupying roles and while maintaining the conditions that ought to hold. This general formulation fits the engineering of new systems; when legacy systems exist, one starts from more information (e.g., roles already in the legacy system).

The select function in **T2** is as follows.

Definition 6.3. The **select function** is defined as follows:

$$\mathsf{Select}(\mathbf{x}, \Pi) \stackrel{def}{=} \{\phi \mid \phi \text{ has the modality } \mathbf{x} \text{ or } \phi \in \Pi^{\rightarrow}\}. \tag{6.46}$$

The part "or $\phi \in \Pi^{\rightarrow}$" ensures that the output of the select function always includes all axioms. A modality can be a core modality or an agency modality. ■

To simplify notation, the following abbreviations are used:

$$\mathsf{Select}(\mathbf{x}, \Pi) \equiv \Pi_{\mathbf{x}}, \tag{6.47}$$

$$\bigcup_{i=1}^{n} \Pi_{\mathsf{R}_i} \stackrel{def}{=} \Pi_{\mathsf{R}}, \tag{6.48}$$

$$\bigcup_{i=1}^{n} \Pi_{\mathsf{A}_i} \stackrel{def}{=} \Pi_{\mathsf{A}}. \tag{6.49}$$

E.g., Π_{R_i} is the set of all expressions where a requirement is assigned to the role R_i. Π_{O} is the set of all expressions which fit the pattern $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_j)$.

Given a ϕ, $\mathsf{Op}(\phi)$ should return all operationalizations of ϕ. The subset relations $\subseteq_\tau$ and $\subset_\tau$ defined for **T1** apply in **T2**.

Definition 6.4. The **operationalization function**

$$\mathsf{Op} : \Delta \longrightarrow \wp(\wp(\Delta)) \tag{6.50}$$

is such that $\Pi \in \mathsf{Op}(\phi \in \Delta)$ if and only if:

1. $\Pi \subseteq_\tau \Delta$, i.e., Π is a subset of Δ and includes all axioms,
2. $\Pi \not\vdash_{\mathbf{2}} \bot$, i.e., Π is consistent,
3. $\Pi \vdash_{\mathbf{2}} \phi$, i.e., ϕ can be deduced from Π,
4. $\nexists \Phi \subset \Pi$, $\Phi \vdash_{\mathbf{2}} \phi$, i.e., Π is minimal,
5. $\forall \psi \in \Pi \setminus \Pi^{\rightarrow}$ s.t. $\psi \notin (\Delta_{\mathsf{R}} \cup \Delta_{\mathsf{A}} \cup \Delta_{\mathsf{cA}} \cup \Delta_{\mathsf{O}})$:
   (a) $\exists \mathsf{R}_i \psi \in \Pi_{\mathsf{R}}$, i.e., there is a role R_i responsible for ψ,
   (b) $\exists \mathsf{A}_j \psi \in \Pi_{\mathsf{A}}$, i.e., there is an agent A_j capable of ψ,
   (c) $\exists \mathsf{cA}_j \psi \in \Pi_{\mathsf{cA}}$, i.e., there is an agent A_j who commits to ψ
   (d) and $\exists \mathsf{O}(\mathsf{R}_i, \mathsf{A}_j) \in \Pi_{\mathsf{O}}$, i.e., R_i is occupied by A_j.

A $\Pi \subseteq_\tau \Delta$ is an operationalization of ϕ if it is consistent, can be used to deduce ϕ, and includes only the necessary information to deduce ϕ. In addition, the fifth condition requires that every member ψ of Π which is *neither* an axiom *nor* a responsibility, ability, commitment, or occupancy relation, must be part of the responsibilities of some role, and there must be an agent who occupies the role, and is able and commits to satisfy, maintain, and/or execute ψ. ■

Below is a comparison of Op in **T2** and Op in **T1**:

- Operationalization in **T2** also requires that $\Pi \in \mathsf{Op}(\phi)$ be consistent, that it is enough to deduce ϕ, and that it is minimal.
- Op in **T2** allows members of $\Pi \setminus \Pi^{\rightarrow}$ to be goals, and not only domain assumptions and tasks as in **T1**. As mentioned above, the reason is that there may be agents able to satisfy goals; if such agents are responsible through roles to satisfy such goals, there is no need to know the tasks and domain assumptions. Operationalization in **T1** was blind to such cases, where how a goal is satisfied remains hidden.[5] Delegating a goal to an agent via a role is thus an acceptable way to operationalize that goal.
- If Π is an operationalization of ϕ in **T1**, then it can be converted into an operationalization of ϕ in **T2** by adding roles responsible for members of $\Pi \setminus \Pi^{\rightarrow}$, agents able to and who commit to satisfy, maintain, or execute members of $\Pi \setminus \Pi^{\rightarrow}$, and having these agents occupy the roles.
- For illustration of Definition 6.4, consider the following sets, all three of which are operationalizations of ϕ:
  - $\{\phi, \mathsf{R}\phi, \mathsf{A}\phi, \mathsf{cA}\phi, \mathsf{O}(\mathsf{R}, \mathsf{A})\} \cup \Delta^{\rightarrow}$: R is responsible for ϕ, agent is able to ϕ, agent commits to ϕ, and A occupies R;
  - $\{\mathbf{g}(p_1), \mathbf{t}(p_2), \mathbf{g}(p_1) \wedge \mathbf{t}(p_2) \rightarrow \phi\}$, to which responsibility is added $\{\mathsf{R}_1\mathbf{g}(p_1), \mathsf{R}_2\mathbf{t}(p_2)\}$, ability $\{\mathsf{A}_1\mathbf{g}(p_1), \mathsf{A}_2\mathbf{t}(p_2)\}$, commitment $\{\mathsf{cA}_1\mathbf{g}(p_1), \mathsf{cA}_2\mathbf{t}(p_2)\}$, occupancy $\{\mathsf{O}(\mathsf{R}_1, \mathsf{A}_1), \mathsf{O}(\mathsf{R}_2, \mathsf{A}_2)\}$, and finally, axioms $\Delta^{\rightarrow}$: ϕ is refined onto $\mathbf{g}(p_1)$ and $\mathbf{t}(p_2)$, and each of the latter is the responsibility of roles occupied by agents who are able to $\mathbf{g}(p_1)$ and $\mathbf{t}(p_2)$.
  - Same as above, but replace $\mathbf{g}(p_1)$ with $\mathbf{t}(p_1)$: ϕ is operationalized via two tasks, so the operationalization follows the pattern from **T1** of having tasks (or domain assumptions) among non-axioms.

The main difference between the solution concepts in **T1** and **T2** comes from the differences in the operationalization functions of the two formalisms. Otherwise, the solution concept in **T2** still requires Consistency and Achievement.

Definition 6.5. A **solution** to the requirements problem given by a requirements database $\Delta \subseteq \mathcal{L}_{\mathbf{2}}$ is a set $S \subseteq_{\tau} \Delta$ which satisfies the following two properties:

1. *Consistency*: $S \not\vdash_{\mathbf{2}} \bot$;
2. *Achievement*: $\forall \phi \in \Delta_{\mathbf{g}}$ s.t. $\nexists \psi,\ \phi \in \bigcup \mathsf{Op}(\psi)$ and $\exists \Pi \in \mathsf{Op}(\phi)$ s.t. $\Pi \subseteq S$.

■

[5] One can obviously change in **T1** the modality of a goal to a task, and thus abstain from having to operationalize what was initially a goal. This trick still misses the information about responsibility and ability that **T2** adds.

6.4 Derived Relations

The inference and conflict relations, and all their specializations defined for **T1** are carried over to **T2**, with the only change that in the definition of each, $\mathbin{\sim_2}$ replaces $\mathbin{\sim_1}$. There is no need to repeat those definitions here.

This section develops relations which include information on agents, roles, and the responsibility, ability, commitment, and occupancy relations.

6.4.1 Dependency Relations

Let R_1 denote the Control Assistant role in the manual LAS, and let $\mathbf{g}(p_1)$ be such that $\mathsf{R}_1\mathbf{g}(p_1)$ and p_1 is for *Acquire incident details*. The acquired incident details are then passed onto another assistant, denote it R_2, who has the goal $\mathbf{g}(p_2)$, where p_2 is for *Allocate incident details*, so that $\mathsf{R}_2\mathbf{g}(p_2)$.

Any agent, say A_2 who occupies R_2 will be able to achieve $\mathbf{g}(p_2)$ only if the goal $\mathbf{g}(p_1)$ is achieved. This can be formalized by asking that $\mathsf{A}_2\mathbf{g}(p_2)$ be deduced if $\mathbf{g}(p_1)$ can be deduced, so that there ought to be $\mathbf{k}(\mathbf{g}(p_1) \to \mathsf{A}_2\mathbf{g}(p_2))$ in the requirements database.

The pattern $\mathbf{k}(\phi \to \mathsf{A}_i\psi)$, where $\phi, \psi \in \Delta \setminus \Delta^{\to}$ is used to define the dependency relation, as follows.

Definition 6.6. There is a **dependency relation** from an agent A_i to $\Pi \subseteq \Delta \backslash \Delta^{\to}$ iff $\mathbf{k}(\bigwedge \Pi \to \mathsf{A}_i\phi) \in \Delta^{\to}$, i.e., if A_i is able to ψ if $\bigwedge \Pi$. ■

Given a dependency $\phi \to \mathsf{A}_i\psi$, if there is also $\mathsf{R}_j\phi$, then it can be said that A_i depends on R_j. Several kinds of the dependency relation can be defined depending on the information present in and absent from Δ.

Table 5 defines the specializations of the dependency relation.

Table 5: Specialization of the dependency relation; replacing ϕ with $\bigwedge \Pi$ is straightforward, and is avoided to simplify notation in the table.

Relation	*Definition*
A_i depends on ϕ	$\mathbf{k}(\phi \to \mathsf{A}_i\psi) \in \Delta^{\to}$.
A_i depends on R_j	$\mathbf{k}(\phi \to \mathsf{A}_i\psi) \in \Delta^{\to}$ and either $\mathsf{R}_j\phi \in \Delta$, or $\phi \equiv \mathsf{R}_j\gamma$.
A_i depends on A_j	$\mathbf{k}(\phi \to \mathsf{A}_i\psi) \in \Delta^{\to}$ and either $\mathsf{A}_j\phi \in \Delta$, or $\phi \equiv \mathsf{A}_j\gamma$, or $\mathsf{cA}_j\phi \in \Delta$, or $\phi \equiv \mathsf{cA}_j\gamma$.
R_i depends on ϕ	$\mathbf{k}(\phi \to \mathsf{A}_i\psi) \in \Delta^{\to}$ and $\mathsf{R}_i\psi \in \Delta$ and $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_i) \in \Delta$.
R_i depends on A_j	$\mathbf{k}(\phi \to \mathsf{A}_i\psi) \in \Delta^{\to}$ and $\mathsf{R}_i\psi \in \Delta$ and $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_i) \in \Delta$ and either $\mathsf{A}_j\phi \in \Delta$, or $\phi \equiv \mathsf{A}_j\gamma$, or $\mathsf{cA}_j\phi \in \Delta$, or $\phi \equiv \mathsf{cA}_j\gamma$.
R_i depends on R_j	$\mathbf{k}(\phi \to \mathsf{A}_i\psi) \in \Delta^{\to}$ and $\mathsf{R}_i\psi \in \Delta$ and $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_i) \in \Delta$ and either $\mathsf{R}_j\phi \in \Delta$, or $\phi \equiv \mathsf{R}_j\gamma$.

In i-star [37], the *dependency relation* can be between roles, between agents, and between agents and roles. The dependency relation in **T2** formalizes and generalizes the i-star dependency relation by allowing an agent/role to depend on (i) one or more requirements which are not tied yet to agents and/or roles, (ii)

one or more agents (e.g., for the case of two agents, replace ϕ with, e.g., $\phi_1 \wedge \phi_2$ and have two agents able, respectively, to ϕ_1 and ϕ_2), (iii) one or more roles.

6.4.2 Conditional Responsibility Relations

The conditional responsibility relation relies on the axiom pattern $\mathsf{k}(\phi \rightarrow \mathsf{R}_i\psi)$. It indicates that R_i becomes responsible for ψ when ϕ. The relation can be used to indicate that the assignment of responsibility depends on the satisfaction of requirements: e.g., if the Resource Allocator role R_i in LAS becomes responsible for the goal $\mathbf{g}(p_1)$, p_1 for *Phone ambulances in stations* when $\mathsf{k}(p_2)$, p_2 for *Dispatch center overloaded*, then there is a conditional responsibility relation $\mathsf{k}(\mathsf{k}(p_2) \rightarrow \mathsf{R}_i\mathbf{g}(p_1))$.

The conditional responsibility relation is defined by analogy to the dependency relation.

Definition 6.7. There is a **conditional responsibility relation** from $\Pi \subseteq \Delta \setminus \Delta^{\rightarrow}$ to a role R_i iff $\mathsf{k}(\bigwedge \Pi \rightarrow \mathsf{R}_i\phi) \in \Delta^{\rightarrow}$, i.e., if R_i is responsible for ψ if $\bigwedge \Pi$. ■

The conditional responsibility relation can be specialized in the obvious way by changing the cardinality of Π and the modalities of its members, as the syntax of **T2** allows.

6.4.3 Commitment Relations

An agent can commit to a role, or to another agent, which gives two kinds of the commitment relation.

Definition 6.8. Agent A_i commits to agent $\mathsf{A}_j \neq \mathsf{A}_i$ if and only if $\mathsf{cA}_i\phi \in \Delta$ and $\mathsf{k}(\phi \rightarrow \mathsf{A}_j\psi) \in \Delta^{\rightarrow}$, i.e., if A_j depends on ϕ and A_i commits to ϕ. ■

Definition 6.9. Agent A_i commits to role R_j if and only if $\exists(\Pi, \psi) \in \mathsf{Arg}(\Delta)$ such that $\phi \in \Pi$ and $\mathsf{R}_j\psi \in \Delta$, i.e., there is an argument for ψ, ϕ is in the premises of that argument, and there is a role responsible for ψ. ■

Singh has argued that commitment should be considered a primitive relation when used for the modeling and reasoning about multi-agent systems (e.g., [29]). His commitment relation is a tuple of the form $C(x, y, G, p)$, read "agent x commits on proposition p to agent y in context G". Both Singh's commitment and cA are primitive relations, and both require agents, not roles. The information in Δ which corresponds to a four-place commitment relation is the following:

$$C(x, y, G, \phi) \equiv \{\mathsf{cA}_x\phi, \mathsf{k}(\phi) \rightarrow \mathsf{A}_y\psi\} \cup G \subseteq \Delta \tag{6.51}$$

$$\text{and } \{\mathsf{cA}_x\phi, \mathsf{k}(\phi) \rightarrow \mathsf{A}_y\psi\} \cup G \not\vdash_{\mathbf{2}} \bot. \tag{6.52}$$

Observe the following:

- Singh's context G is not explicitly a set of expressions, but a group of agents occupying roles. This is not an issue since **T2** relates requirements to agents via responsibility assignments to roles, whereby roles are occupied by agents. In other words, Singh's group is here a set of responsibility, occupancy, and ability relations.

- A major difference between Singh's commitment relation and commitment in **T2** is that the former allows a commitment to be about other commitments: it is thus possible to say that an agent commits to assign a commitment to another agent. Since **T2** does not allow a commitment to appear as a parameter in cA, a commitment in **T2** cannot refer to another commitment.

6.4.4 Delegation Relations

Suppose a role R_1 is responsible for the goal $\mathbf{g}(p)$, and that the goal is refined onto two tasks, $\mathbf{t}(q_1)$ and $\mathbf{t}(q_2)$, so that $\mathbf{k}(\mathbf{t}(q_1) \wedge \mathbf{t}(q_2) \rightarrow \mathbf{g}(p)) \in \Delta^{\rightarrow}$. If the role R_1 is *not* responsible for both of these tasks, then there is a delegation relation from R_1 to the roles responsible for these two tasks. If $\mathsf{R}_2\mathbf{t}(q_1)$ and $\mathsf{R}_3\mathbf{t}(q_2)$ are both in Δ, then the delegation relation is from R_1 to the two roles R_2 and R_3.

Definition 6.10. There is a **delegation relation** from a role R_i to a set of roles $\{\mathsf{R}_1, \ldots, \mathsf{R}_{n\geq 1}\}$ if and only if:

1. $\mathsf{R}_i\phi \in \Delta$, i.e., R_i is responsible for ϕ,
2. $\exists(\Pi, \phi) \in \mathsf{Arg}(\Delta)$ s.t. $\forall \mathsf{R}_j \in \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$, $\exists \psi \in \Pi \setminus \Delta^{\rightarrow}$ and $\mathsf{R}_j\psi \in \Delta$, i.e., every role in $\{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$ is responsible for a member of $\Pi \setminus \Delta^{\rightarrow}$;
3. $\exists \mathsf{R}_j \neq \mathsf{R}_i,\ \mathsf{R}_j \in \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$, i.e., at least one of the roles responsible for the members of $\Pi \setminus \Delta^{\rightarrow}$ is different than R_i.

■

Table 6 lists relations obtained by specializing the delegation relation.

Table 6: Specialization of the delegation relation.

Relation	*Definition*
R_i delegates to $\{\mathsf{R}_1, \ldots, \mathsf{R}_{n\geq 1}\}$	Definition 6.10.
R_i delegates goal to $\{\mathsf{R}_1, \ldots, \mathsf{R}_{n\geq 1}\}$	Definition 6.10 and $\phi \equiv \mathbf{g}(p)$.
R_i delegates task to $\{\mathsf{R}_1, \ldots, \mathsf{R}_{n\geq 1}\}$	Definition 6.10 and $\phi \equiv \mathbf{t}(p)$.
R_i delegates $\mathbf{k}$ to $\{\mathsf{R}_1, \ldots, \mathsf{R}_{n\geq 1}\}$	Definition 6.10 and $\phi \equiv \mathbf{k}(p)$.

Delegation generates dependency relations. When there is a delegation from a role to other roles, the agent occupying the former depends on agents occupying the latter. This can be formalized with the following macro on the requirements database:

Definition 6.11. The **dependency from delegation macro** is the defined as

follows:

If

$\mathsf{R}_i\phi \in \Delta$ and

$\exists(\Pi, \phi) \in \mathsf{Arg}(\Delta),\ \forall \mathsf{R}_j \in \{\mathsf{R}_1, \ldots, \mathsf{R}_n\},\ \exists \psi \in \Pi \setminus \Delta^{\rightarrow},\ \mathsf{R}_j\psi \in \Delta$ and

$\exists \mathsf{R}_j \neq \mathsf{R}_i,\ \mathsf{R}_j \in \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$ and

$\mathsf{O}(\mathsf{R}_i, \mathsf{A}_i) \in \Delta$

then

Add $\mathbf{k}(\bigwedge(\Pi \setminus \Pi^{\rightarrow}) \rightarrow \mathsf{A}_i\phi)$ to Δ.

In words, if (i) R_i is responsible for ϕ, and (ii) Π is sufficient to deduce ϕ, and (iii) there are roles responsible for members of $\Pi \setminus \Delta^{\rightarrow}$, and (iv) at least one of these roles is different from R_i, and (v) agent A_i occupies R_i, then add the dependency of A_i on $\Pi \setminus \Pi^{\rightarrow}$. ■

The macro adds the minimal condition for the presence of a dependency relation. It is then straightforward to determine which kind of dependency results from the delegation, based on the definitions of the various specialization relations in Table 5.

6.4.5 Inference Relations

The inference relation and its associated argument concept remain essentially the same in **T2** as in **T1**, except for the obvious change in definitions from $\mid\!\sim_{\mathtt{1}}$ to $\mid\!\sim_{\mathtt{2}}$.

The relationships on roles and agents in **T2** result in revised goal refinement, task decomposition, and goal operationalization (and means-ends) relations. These differences are visible in Table 7, and are as follows:

- Every requirement in a refinement of a goal should be the responsibility of a role;
- Roles responsible for requirements in a refinement should all be occupied by agents;
- Agents who occupy the said roles should be able to discharge the responsibilities of these roles.

To simplify the definition of the said relations, the operational argument concept is introduced as follows.

Definition 6.12. The argument $(\Pi, \phi) \in \mathsf{Arg}(\Delta)$ is an operational argument if and only if:

1. $\forall \psi \in \Pi \setminus \Delta^{\rightarrow}$, $\exists \mathsf{R}_i\psi \in \Delta$, i.e., there is a role responsible for every requirement in the premises of the argument;
2. $\forall \psi \in \Pi \setminus \Delta^{\rightarrow}$, $\exists \mathsf{A}_j\psi \in \Delta$, i.e., there is an agent able to satisfy every requirement in the premises of the argument;

3. $\forall \psi \in \Pi \setminus \Delta^{\rightarrow}$, if $\mathsf{R}_i\psi \in \Delta$ and $\mathsf{A}_j\psi \in \Delta$, then $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_j) \in \Delta$, i.e., the agent able to satisfy a requirement in the premises of the argument occupies the role responsible for the satisfaction of the requirement.

■

Table 7: Specialization of the inference relation.

Relation	*Definition*
Goal refinement of $\mathbf{g}(p)$	(i) $\exists(\Pi, \mathbf{g}(p)) \in \mathsf{Arg}(\Delta)$ and (ii) $\Pi \setminus \Delta^{\rightarrow} \subseteq \Delta_{\mathbf{g}}$ and (iii) $(\Pi, \mathbf{g}(p))$ is an operational argument.
Task decomposition of $\mathbf{t}(p)$	$\exists(\Pi, \mathbf{t}(p)) \in \mathsf{Arg}(\Delta)$ and (ii) $\Pi \setminus \Delta^{\rightarrow} \subseteq \Delta_{\mathbf{g}} \cup \Delta_{\mathbf{t}}$ and (iii) $(\Pi, \mathbf{g}(p))$ is an operational argument.
Goal operationalization of $\mathbf{g}(p)$	$\exists(\Pi, \mathbf{g}(p)) \in \mathsf{Arg}(\Delta)$ and (ii) $\Pi \setminus \Delta^{\rightarrow} \subseteq \Delta_{\mathbf{t}}$ and (iii) $(\Pi, \mathbf{g}(p))$ is an operational argument.
Means-ends, with $\mathbf{g}(p)$ as "ends"	*Same as* Goal operationalization.

6.4.6 Conflict Relations

The conflict relation in **T1** required that two conditions be satisfied for the members of a set $\Pi \setminus \Delta^{\rightarrow}$ to be in conflict: (i) there must be an axiom, in which the antecedent is the conjunction of all members of $\Pi \setminus \Delta^{\rightarrow}$, and the consequent of which is $\bot$, and (ii) the set $\Pi \setminus \Delta^{\rightarrow}$ should be minimal, i.e., no subset thereof is itself inconsistent.

The two conditions, inconsistency and minimality remain applicable in **T2**. The conflict relation is defined in the same way as in **T1**, except that $\mid\!\sim_{\mathbf{1}}$ is replaced with $\mid\!\sim_{\mathbf{2}}$. Moreover, specialization to Type-A, Type-B, and Type-C remains applicable, again with the exception that $\mid\!\sim_{\mathbf{1}}$ gets replaced with $\mid\!\sim_{\mathbf{2}}$. This is summarized in the first four lines of Table 8. The remainder of the Table lists further and straightforward specializations of the conflict relations. Note that the specializations onto Type-A, -B, and -C do partition the extension of the Conflict relation. However, the specialization of Type-A onto Responsibility, Ability, Occupancy, and Commitment Conflict is obviously not a partition of the extension of Type-A. Rather, these cover interesting and homogeneous parts of that extension: what is missing for example is a Type-A relation where members of an alternative do not all have the same modality.

6.5 Database Interface

The database interface for $\Delta \subseteq \mathcal{L}_{\mathbf{2}}$ needs all filters defined for **T1**. The notion of a proto-solution is still relevant in **T2**, as are filters defined for goals, tasks, domain assumptions, and conflicts. Filters added here apply to the new relations in **T2**. Filters are as follows; in every filter, $\mathbf{y} \in \{\mathsf{R}, \mathsf{A}, \mathsf{cA}, \mathsf{O}(\mathsf{R}, \mathsf{A})\}$:

Table 8: Specialization of the conflict relation. For each conflict relation, members of $\{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in conflict.

Relation	*Definition*
Conflict	(i) $\exists \mathbf{k}((\bigwedge_{i=1}^{n\geq 1} \phi_i) \rightarrow \bot) \in \Delta$, and (ii) $\nexists \Pi \subset \{\phi_1, \ldots, \phi_{n\geq 1}\}$ s.t. $\Pi \cup \Delta^{\rightarrow} \vdash_{\mathbf{2}} \bot$.
Type-A Conflict	(i) $\{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in conflict, and (ii) $\mathsf{Alt}(\Pi) \geq 2$.
Type-B Conflict	(i) $\{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in conflict, and (ii) $\mathsf{Alt}(\Pi) = 1$.
Type-C Conflict	(i) $\{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in conflict, and (ii) $\mathsf{Alt}(\Pi) = 0$.
Specialization of Type-A Conflict:	
Responsibility Conflict	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-A conflict, and (ii) every alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is a responsibility relation (i.e., fits the pattern $\mathsf{R}_i\phi$).
Ability Conflict	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-A conflict, and (ii) every alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is an ability relation (i.e., fits the pattern $\mathsf{A}_i\phi$).
Occupancy Conflict	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-A conflict, and (ii) every alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is an occupancy relation (i.e., fits the pattern $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_j)$).
Commitment Conflict	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-A conflict, and (ii) every alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is a commitment relation (i.e., fits the pattern $\mathsf{cA}\phi$).
Specialization of Type-B Conflict:	
Blocked Responsibility	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-B conflict, and (ii) the alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is a responsibility relation (i.e., fits the pattern $\mathsf{R}_i\phi$).
Blocked Ability	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-B conflict, and (ii) the alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is an ability relation (i.e., fits the pattern $\mathsf{A}_i\phi$).
Blocked Occupancy	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-B conflict, and (ii) the alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is an occupancy relation (i.e., fits the pattern $\mathsf{O}(\mathsf{R}_i, \mathsf{A}_j)$).
Blocked Commitment	(i) $\Phi = \{\phi_1, \ldots, \phi_{n\geq 1}\}$ are in Type-B conflict, and (ii) the alternative Ψ^{Φ} from Φ is such that every member of $\Psi^{\Phi} \setminus \Delta^{\rightarrow}$ is a commitment relation (i.e., fits the pattern $\mathsf{cA}_i\phi$).

- All requirements with core modality $\mathsf{x} \in \{\mathbf{g}, \mathbf{k}, \mathbf{t}\}$ which are not in a responsibility, or ability, or commitment, or occupancy relation:

$$\mathsf{ANo}(\mathsf{x}, \mathbf{y}, \Delta) = \{\phi \in \Delta_{\mathsf{x}} \mid \nexists \mathbf{y}\phi \in \Delta\}. \tag{6.53}$$

- All instances of a responsibility, or ability, or commitment, or occupancy relation, which are conclusions of n arguments:

$$\mathsf{AArgConcl}(n, \mathbf{y}, \Delta) = \{\phi \in \Delta_{\mathbf{y}} \mid \exists X \subseteq \mathsf{Arg}(\Delta),\ \forall(\Pi, \psi) \in X, \\ \psi \equiv \phi \text{ and } |X| = n\}. \tag{6.54}$$

- All instances of a responsibility, or ability, or commitment, or occupancy relation, which are neither conclusions of arguments, nor consequents in axioms:

$$\mathsf{ABasic}(\mathbf{y}, \Delta) = \mathsf{AArgConcl}(0, \mathbf{y}, \Delta) \setminus \{\phi \mid \mathsf{AxiomsCons}(\phi)\}. \tag{6.55}$$

- All instances of a responsibility, or ability, or commitment, or occupancy relation, which have n operationalizations:

$$\mathsf{AOp}(n, \mathbf{y}, \Delta) = \{\phi \in \Delta_{\mathbf{y}} \mid \exists X \subseteq \mathsf{Arg}(\Delta),\ \forall(\Pi, \psi) \in X, \\ \psi \equiv \phi \text{ and } |X| = n\}. \tag{6.56}$$

- All instances of a responsibility, or ability, or commitment, or occupancy relation, which participate in premises of at least n arguments:

$$\mathsf{AArgPrem}(n, \mathbf{y}, \Delta) = \{\phi \in \Delta_{\mathbf{y}} \mid \exists X \subseteq \mathsf{Arg}(\Delta),\ \forall(\Pi, \psi) \in X \\ \phi \in \Pi \text{ and } |X| = n\}. \tag{6.57}$$

A filter can be defined for every relation introduced for **T2**, in order to find all instances of that relation. E.g., a filter can be defined to return all dependent agents, i.e., every agent A_j such that there is $\mathsf{k}(\bigwedge_i^{n \geq 1} \phi_i \rightarrow \mathsf{A}_j\psi)$. Defining such filters is straightforward, given the definitions of the relations (cf., §6.4).

6.6 Discussion

The responsibility, ability, occupancy, and commitment relations allowed both for the introduction of agents and roles, and for a deeper specialization of the inference and conflict relations. Common relations such as goal refinement, task decomposition, and goal operationalization obtained new conditions (through the operational argument concept), pertaining to role responsibilities, and agents' abilities, commitments, and their assignment to roles.

In contrast to i-star, the main formalism in RE which focuses on agents and roles, **T2** formalizes and generalizes its relations. **T2** also extends the set of relations found in i-star by adding the conflict relation and its specializations.

7 T3

T3 adds the soft counterparts to the Goal and Domain assumption concepts, and the preference relation to **T2**. The effects are as follows:

- In **T1** and **T2**, no information could be made explicit for *comparison of solutions in terms of desirability*. The preference relation is a binary relation which indicates the relative desirability of satisfying either of two expressions. E.g., if there is a preference relation saying that ϕ is strictly preferred to ψ, then this reads that satisfying ϕ is strictly more desirable than satisfying ψ.

- Every instance of the preference relation acts as a criterion for the comparison of alternative solutions. If ϕ is strictly preferred to ψ, and solution A satisfies ϕ, while solution B satisfies ψ, then the preference suggests that A dominates B over this criterion alone.

- While the Softgoal concept has had various definitions in RE, its instances are usually vague statements about quality expected from the system-to-be: e.g., "Privacy should be maintained", "System should be reliable", "Queries should be answered quickly", and so on. These statements are vague because they include or refer to adjectives, such as "private", "reliable", "quick", which have no universal definition. If there was such a definition, it would be clear when such adjectives apply, and thereby when a softgoal is satisfied. Instead, a softgoal acts as a macro which generates preference relations: e.g., let ϕ and ψ be two requirements, such that it is known that satisfying ϕ makes the system-to-be more reliable than if ψ is satisfied. The softgoal "System should be reliable" generates a preference relation according to which ϕ is strictly preferred to ψ. By generating preference relations, a softgoal acts as a question-answering mechanism: given a softgoal, and two options which can be compared in terms of the measure referred to in the softgoal, the softgoal says which of the options is more desirable. Soft domain assumptions is used in an analogous way.

- Comparison of solutions allows the ranking of solutions. Ranking is established using *decision rules*, which use preference relations. A decision rule gives a weight to each preference relation, and thereby suggests which preferences are more important than others. E.g., a decision rule may make all preferences equally important, and thus suggest that the solution which satisfies the highest number of preferred requirements ranks highest. It should be clear that every Techne formalism has a decision rule, even when preferences (and softgoals) are absent: e.g., **T1** says that any instance of the solution concept can be chosen, and this in itself is a decision rule, even if simplistic. When preferences are added, more interesting decision rules can be defined.

7.1 Illustration

The Softgoal concept adds a modality to the core modalities from **T2**, and is denoted **sg**. Suppose that there are the following two softgoals for LAS:

$\mathbf{sg}(\tilde{w}_1$: Make few errors when identifying ambulances);

$\mathbf{sg}(\tilde{w}_2$: Quickly establish if two or more incidents are reported for the same location);

The two propositions use gradable adjectives "few" and "quick" which are vague, hence the classification of these propositions as softgoals.

The number of errors made when identifying ambulances depends on how Control assistants, denote this role R_1, are informed about which ambulances are allocated to which incidents. Let R_2 denote the role of the dispatch interface. There are roughly two ways to operationalize the goal $\mathbf{g}(p_1$: Identify available ambulances), which is the responsibility of R_1, i.e., $\mathsf{R}_1\mathbf{g}(p_1) \in \Delta$. One consists of having the role R_2 make available the list of allocated ambulances and update the list every time an ambulance is allocated to an incident; this operationalization involves two tasks:

$\mathbf{t}(u_1$: Show the list of all available and of all allocated ambulances);

$\mathbf{t}(u_2$: Update the list of all available and of all allocated ambulances every time an ambulance is assigned to an incident).

Both are responsibility of R_2, so that $\mathsf{R}_2\mathbf{t}(u_1)$ and $\mathsf{R}_2\mathbf{t}(u_2)$ are also in the requirements database, along with $\mathbf{k}(\mathbf{t}(u_1) \wedge \mathbf{t}(u_2) \rightarrow \mathbf{g}(p_1))$.

Another operationalization of $\mathbf{g}(p_1)$ consists of having R_1 manually keep the list of available ambulances:

$\mathbf{t}(u_3$: Manually keep track of available ambulances).

For this option, the database includes also $\mathbf{k}(\mathbf{t}(u_3) \rightarrow \mathbf{g}(p_1))$ and $\mathsf{R}_1\mathbf{t}(u_3)$.

Experience from similar past systems, or other sources of expertise may suggest that operationalization via $\mathbf{t}(u_1)$ and $\mathbf{t}(u_2)$ is less prone to errors when ambulances are identified, than the operationalization via $\mathbf{t}(u_3)$. Let the former operationalization be labeled P_1, and the latter P_2. Given these two options, the softgoal gives the preference relation $\mathsf{P}_1 \succ \mathsf{P}_2$, which reads "option P_1 is strictly more desirable than option P_2".

The time needed to establish if two or more incidents have been reported for the same location depends on how Control assistants discharge the goal $\mathbf{g}(p_2$: Check if multiple incidents are reported at same location). Consider two operationalizations of $\mathbf{g}(p_2)$:

1. $\mathbf{g}(p_2)$ is first refined by the goal $\mathbf{g}(p_3$: Aid the identification of duplicate calls for same incident) and the responsibility assignment $\mathsf{R}_2\mathbf{g}(p_3)$, so that $\mathbf{k}(\mathbf{g}(p_3) \wedge \mathsf{R}_2\mathbf{g}(p_3) \rightarrow \mathbf{g}(p_2))$. Then, the goal $\mathbf{g}(p_3)$ is operationalized onto two tasks, $\mathbf{t}(u_4$: List of open incident locations incidents visible in the

dispatch interface) and $\mathbf{t}(u_5$: Update the list of open incident locations every time an open incident is added) via $\mathbf{k}(\mathbf{t}(u_4) \wedge \mathbf{t}(u_5) \rightarrow \mathbf{g}(p_3))$. The two tasks are responsibility of R_2, so that also $\mathsf{R}_2\mathbf{t}(u_4) \in \Delta$ and $\mathsf{R}_2\mathbf{t}(u_5) \in \Delta$.

2. $\mathbf{g}(p_2)$ is operationalized by $\mathbf{t}(u_6$: Every control assistant manually keeps track of locations of open incidents), via $\mathbf{k}(\mathbf{t}(u_6) \rightarrow \mathbf{g}(p_2))$, and the responsibility assignment is $\mathsf{R}_2\mathbf{t}(u_6)$.

If it judged that the first option above will result in lower times to identify duplicate incident reports than the second option, then the softgoal generates the preference relation for the former operationalization over the latter. If the former is labeled P_3 and the latter P_4, then the preference relation is $\mathsf{P}_3 \succ \mathsf{P}_4$.

Consider now an instance of the Soft domain assumption concept, that is, a vague domain assumption: $\mathbf{sk}(w_3$: Enough ambulances are operational). When there are options as to how many ambulances must be made operational, then this domain assumption generates preference relations over these options. The options may be alternative operationalizations of a goal that involves choosing the number of ambulances to be bought and serviced at each ambulance center.

7.2 Formalization

Semantic Domain. Two kinds of natural language propositions are distinguished in **T3**: vague and clear (i.e., non-vague) propositions. In **T1** and **T2**, all propositions were treated as clear, regardless of them actually being such according to the criteria introduced below.

A natural language proposition is vague if all of the following conditions verify for that proposition (e.g., [18] and related):

1. *Truth conditional variability.* Whether one would accept this proposition as true depends on the context in which it is used. *Quick system*, *storage is sufficient*, *database has an intuitive structure* are all obviously relative to the context in which they are used.

2. *Existence of borderline cases.* Whatever the context of use, there will be three sets of objects for which the truth of the proposition can be evaluated to true, false, or either. In case of requirements, these objects can be thought of as the system, its environment, or any part of, or combination of (the parts of) these two. One set will be objects which clearly satisfy the condition in the proposition, so the proposition is judged true for them. Another will be such that the condition clearly fails for them. The third set includes the so-called borderline objects, for which it is unclear if the proposition is true or false; in other words, there is no definite criterion which suggests either of the truth values. An object is a *borderline case* if it is difficult or impossible to determine if it satisfies the condition stated in the proposition. Taken out of context, a system with any response time is a borderline case for *quick system*: if it is judged that a system with response time x is quick, and one with $5x$ is clearly not quick, then is $2x$ quick or not?

3. *The Sorites Paradox.* When employed within a particular form of argument, the proposition will give rise to the Sorites paradox (paradox of the heap). The argument is usually laid out as follows (e.g., [16]):

   1 grain does not make a heap.
   If 1 grain does not make a heap then 2 grains do not.
   If 2 grains do not make a heap then 3 grains do not.
   ...
   If 9,999 grains do not make a heap then 10.000 do not.
   10.000 grains do not make a heap.

   The argument above appears valid, premises true, yet the conclusion false. While the argument proceeds by addition, it can be reformulated to proceed by subtraction: start by claiming that 10.000 grains of wheat do make a heap, and subtract grains of wheat to arrive at the undesired conclusion that 1 grain of wheat does make a heap. Same argument, be it by addition or subtraction can be often constructed. If it is claimed that a quick system is one which has a response time of x, one could construct the same argument as above, by adding in each step, say, one milisecond to x.

A vague proposition subsumes a scale and an order over the values on the scale. *Quick* implies a time scale, *big* some sort of size scale, *simple* a scale of simplicity, and so on. Each of these suggests an order of desirability over the values on the scale: e.g., *quick* suggests an order opposite of the order suggested by *slow*.

Distinguishing vague and clear natural language propositions in **T3** results in the Softgoal concept as a vague counterpart to Goal and the Soft domain assumption concept as the vague counterpart to Domain assumption.

The semantic domain in **T3** includes vague and precise natural language propositions, agents, and roles. There are now five kinds of natural language expressions, members of extensions of the Goal, Domain assumption, Task, Softgoal and Soft domain assumption concepts. A proposition is a softgoal if it states a vague desired condition, a soft domain assumption if it states a vague believed condition. There are no vague tasks, because a task requires intention, and it is assumed that intentions are clear enough. **T3** keeps all relations from **T2** and adds two kinds of *preference relation*, strict preference and indifference, between arbitrary sets of expressions formed over propositions and relations.

Syntax. p, q, r, indexed or primed as needed, refer to *clear* natural language propositions. When a natural language proposition is vague, it is denoted with indexed or primed $\tilde{p}, \tilde{q}, \tilde{r}$. A and R are indexed, and denote, respectively, an agent and a role. The symbol **sg** refers to the softgoal modality and **sk** to the soft domain assumption modality.

The language in **T3** is a finite set $\mathcal{L}_{\mathbf{3}}$ of expressions, which is split onto two parts. The *modeling* part, denoted $\mathcal{L}_{\mathbf{3}}^{M}$, equals the language from **T2**, i.e., $\mathcal{L}_{\mathbf{3}}^{M} \equiv \mathcal{L}_{\mathbf{2}}$, and is used to specify information that states the requirements problem and its alternative solutions. The *decision-making* part, denoted $\mathcal{L}_{\mathbf{3}}^{D}$

includes expressions which refer to preference relations, softgoals, and soft domain assumptions, that is, information which is used for the comparison of alternative solutions modeled using $\mathcal{L}_{\mathbf{3}}^{D}$.

The language of **T3** is the union $\mathcal{L}_{\mathbf{3}} = \mathcal{L}_{\mathbf{3}}^{M} \cup \mathcal{L}_{\mathbf{3}}^{D}$ of expressions where every $\phi \in \mathcal{L}_{\mathbf{3}}^{M}$ and every $\tilde{\phi} \in \mathcal{L}_{\mathbf{3}}^{D}$ satisfies the following BNF specification:

$$a ::= \mathbf{g}(p) \mid \mathbf{k}(p) \mid \mathbf{t}(p) \tag{7.58}$$

$$b ::= \mathsf{R}a \mid \mathsf{A}a \mid \mathsf{cA}a \tag{7.59}$$

$$c ::= a \mid b \mid \mathsf{O}(\mathsf{R}, \mathsf{A}) \tag{7.60}$$

$$d ::= (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow c \mid (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow \bot \tag{7.61}$$

$$\phi ::= c \mid \mathbf{k}(d) \tag{7.62}$$

$$e ::= \{c_1, \ldots, c_{n\geq 1}\} \tag{7.63}$$

$$f ::= e \succ e \mid e \approx e \tag{7.64}$$

$$g ::= \{f_1, \ldots, f_{n\geq 0}\} \tag{7.65}$$

$$\tilde{\phi} ::= g \mid \mathbf{sg}(\tilde{p}) \leftarrow g \mid \mathbf{sk}(\tilde{p}) \leftarrow g \mid g \rightarrow \bot \tag{7.66}$$

The first five rules above ensure that $\mathcal{L}_{\mathbf{3}}^{M}$ is identical to $\mathcal{L}_{\mathbf{2}}$, so that all remarks on the versatility and limitations of **T2** syntax apply here as well. Novelty in syntax starts with the rule that generates e, which is a set of requirements or relations on agents and roles, i.e., a set of cs. The set then participates, via f, in a strict preference relation or an indifference relation, to compare in terms of desirability these sets, and thereby allow the comparison of (parts of) solutions to a requirements problem. A softgoal and soft domain assumption generate sets of preference relations, which is reflected in g being a set of preference/indifference relations, which are then related to the softgoal or soft domain assumption in $\tilde{\phi}$. It is allowed for a preference relation, or a set of preference relations not to be generated by a softgoal or soft domain assumption. It is also allowed, when $n = 0$ in g, for a softgoal or soft domain assumption not to have generated preference relations – instances of these concepts can thus be added to a requirements database without having also and at the same time to add preference relations that they generate.

Limitations of the grammar in **T3** are as follows:

- Statements such as $\mathbf{g}(\phi \succ \psi)$ are not expressions. **T3** does not allow preferences to be labeled with a core modality, or to have a role responsible for a preference, or an agent able to satisfy a preference, or have an agent commit to a preference.

- Agents cannot commit or be able to satisfy a softgoal or maintain a soft domain assumption. This limitation is due to the vagueness of these requirements. Because they are vague, it cannot be decided if a softgoal has been satisfied, or a soft domain assumption maintained. Instead, by

generating preferences over clear requirements, softgoals and soft domain assumptions suggest how desirable are clear abilities or commitments of agents.

- For a similar reason, softgoals and soft domain assumptions cannot be responsibilities of roles. Rather, they state how desirable various clear responsibilities are.

- Softgoals and soft domain assumptions cannot be in conflict. This is not because there are no such conflicts, but because all such conflicts should be tolerated. The preferences that they generate can be in conflict, and when such conflict is present, it is understood as a tradeoff. E.g., if the softgoal *Respond quickly to emergency calls* gives the preference $\mathbf{t}(p_1) \succ \mathbf{t}(p_2)$, but the softgoal *Lower the cost of ambulance dispatching* gives the opposite preference $\mathbf{t}(p_2) \succ \mathbf{t}(p_1)$, then there is a conflict $\{\mathbf{t}(p_1) \succ \mathbf{t}(p_2), \mathbf{t}(p_2) \succ \mathbf{t}(p_1)\} \rightarrow \bot$.

The symbol $\leftarrow$ relates a softgoal or soft domain assumption to the preference set that it generates. The set is obtained by the application of the macro obtained by the softgoal. Every softgoal and soft domain assumption macro is of the form, for $\mathsf{x} \in \{\mathsf{sg}, \mathsf{sk}\}$:

$\mathsf{x}(\tilde{p}) \stackrel{def}{=}$ If satisfying ϕ is

- at least as desirable as satisfying ψ according to $\tilde{p}$, then add $\phi \succeq \psi$ to Δ; or is
- strictly more desirable than satisfying ψ according to $\tilde{p}$, then add $\phi \succ \psi$ to Δ; or is
- as desirable as satisfying ψ according to $\tilde{p}$, then add $\phi \approx \psi$ to Δ.

A softgoal and soft domain assumption macro is a function which takes two (potentially singleton) sets of requirements and returns a preference relation between them. The function is defined interactively: the macro is simply an interface which asks the modeler to choose two requirements to compare according to the softgoal or soft domain assumption, and the modeler chooses one of the three options above.

T3 leaves softgoal macros outside its syntax, as there are no interactions between the macros and the language $\mathcal{L}_{\mathbf{3}}$.

Semantic Mapping. The semantic mapping function relates members in $\mathcal{L}_{\mathbf{3}}$ to the objects identified in the semantic domain and the relations between these objects. How this happens is clear from the informal discussions above.

Consequence Relation. The consequence relation $\mid\!\sim_{\mathbf{3}}$ is defined in the same way as $\mid\!\sim_{\mathbf{2}}$ and $\mid\!\sim_{\mathbf{1}}$, even though the formal languages for each of these relations are different.

Definition 7.1. For $\Pi \subseteq_\tau \mathcal{L}_{\mathbf{3}}$ and $\phi \in \mathcal{L}_{\mathbf{3}}$, the **consequence relation** $\mid\!\sim_{\mathbf{3}}$ is such that:

- $\Pi \mid\!\sim_{\mathbf{3}} \phi$ if $\phi \in \Pi$, or
- $\Pi \mid\!\sim_{\mathbf{3}} x$ if $\forall 1 \leq n$, $\Pi \mid\!\sim_{\mathbf{3}} \phi_i$ and $\mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow x) \in \Pi$,

and $\mid\!\sim_{\mathbf{3}}$ satisfies the axioms:

$$\forall \Pi_i, \Pi_j, \ \{\Pi_i \succ \Pi_j, \Pi_j \succ \Pi_i\} \rightarrow \bot, \tag{7.67}$$

$$\forall \Pi_i, \Pi_j, \ \{\Pi_i \succeq \Pi_j, \Pi_j \succ \Pi_i\} \rightarrow \bot. \tag{7.68}$$

The two axioms indicate that contradictory preferences must produce inconsistency. ■

Remarks on soundness and completeness made for $\mid\!\sim_{\mathbf{1}}$ and $\mid\!\sim_{\mathbf{2}}$ apply for $\mid\!\sim_{\mathbf{3}}$.

7.3 Problem & Solution Concepts

Adding preferences results in a significant departure of the problem and solution concepts in **T3** compared to those in **T1** and **T2**. The major change is that now, there is a large number of alternative requirements problem concepts. Differences between them are due to each using a particular *decision rule*.

A decision rule uses preference relations to produce a ranking of alternative solutions. Problem definitions in **T1** and **T2** subsumed a single decision rule: choose *any* solution. Given preferences, solutions can be compared, and a decision rule will state how to produce a ranking of solutions from preferences. The requirements problem *template* is then as follows.

Definition 7.2. Given a set of goals, domain assumptions, and agents, find the solution which ranks highest according to the chosen decision rule. ■

The chosen decision rule is a macro, which results in **T3** having any number of macros, as many as one may care to specify. One such macro was given in Example 3.5 earlier.

While decision rules can differ considerably, none should single out a Pareto dominated solution. Suppose that there are two solutions, S_1 and S_2. S_1 Pareto dominates S_2 if and only if it is possible to obtain S_2 by removing requirements from S_1, while still keeping S_2 a solution.

The select and operationalization functions remain the same as in **T2**, with the obvious difference that $\mid\!\sim_{\mathbf{2}}$ gets replaced with $\mid\!\sim_{\mathbf{3}}$ and that the requirements database now has two parts, reflecting the partition of $\mathcal{L}_{\mathbf{3}}$.

The requirements database is split onto modeling and decision information, that is, $\Delta \subseteq \mathcal{L}_{\mathbf{3}}$ is such that $\Delta = \Delta^M \cup \Delta^D$, $\Delta^M \subseteq \mathcal{L}^M_{\mathbf{3}}$ and $\Delta^D \subseteq \mathcal{L}^D_{\mathbf{3}}$.

As preferences compare solutions, they are not part of solutions. Solutions are identified in Δ^M, which obeys the grammar of **T2**, so that the solution concept is analogous to the solution concept in **T2**, in that it requires that the same two conditions, Consistency and Achievement, be satisfied.

7.4 Derived Relations

Since $\mathcal{L}_{\mathbf{2}} \subseteq \mathcal{L}_{\mathbf{3}}^{M}$, all relations from **T2** apply in **T3**, pending obvious changes to their definitions (e.g., replacing $\mid\!\sim_{\mathbf{2}}$ with $\mid\!\sim_{\mathbf{3}}$).

This section develops relations which involve preferences and vague requirements.

7.4.1 Contribution Relations

The NFR framework [23] models nonfunctional requirements as goals which can be satisfied to different levels, rather than treating them as two-valued. Every nonfunctional goal serves as a criterion for the comparison of alternative designs of the system-to-be. Each alternative will satisfy each nonfunctional goal to some extent. All alternatives can then be compared over each criterion by contrasting the satisfaction levels reached by every alternative on every nonfunctional goal.

The contribution relation in NFR relates a (part of) a design alternative to a nonfunctional goal. The relation carries additional information in the form of a contribution value, which qualifies how well the alternative satisfies the nonfunctional goal. It is a triple $\mathsf{Contribute}(a, v, s)$, where s is a softgoal, a a set of requirements, and v a contribution value.

To see how contribution relations can be captured using **T3**, observe firstly that a nonfunctional goal is a softgoal. Since a nonfunctional goal can be *more* or *less* satisfied, it is a vague requirement. Secondly, a contribution relation from an alternative to a nonfunctional goal is useful only if there is another alternative and another contribution relation from that other alternative to the same nonfunctional goal. I.e., contribution value on a contribution relation needs to be interpreted relative to contribution values on other contribution relations targeting the same nonfunctional goal.

The definition of contribution relations using **T3** is as follows.

Definition 7.3. Let $\mathbf{sg}(\tilde{p})$ be a softgoal such that $\mathbf{sg}(\tilde{p}) \leftarrow \{\Pi_i \succ \Pi_j \mid 1 \leq i, j \leq n,\ n \geq 2,\ i \neq j\}$, i.e., $\mathbf{sg}(\tilde{p})$ generates n preference relations. Let there be two possible contribution values, denoted + and ++. There is a ++ contribution relation from Π_i to $\mathbf{sg}(\tilde{p})$ and a + contribution relation from Π_j to $\mathbf{sg}(\tilde{p})$ if and only if $\Pi_i \succ \Pi_j$. ■

The important property of this definition is that it changes as the number of contribution values changes, and as the number of alternative sets Pi_i changes. The simplest case is when there are two alternatives, Π_1 and Π_2, and there is a strict preference relation between them, say $\Pi_1 \succ \Pi_2$. If this preference is generated by some softgoal $\mathbf{sg}(\tilde{p})$, then there are two contribution relations, one from Π_1 to $\mathbf{sg}(\tilde{p})$, and another from Π_2 to $\mathbf{sg}(\tilde{p})$. The former must carry a higher contribution value than the latter, to reflect the preference relation. When the number of alternatives is higher, two cases ought to be distinguished:

- The strict preference relations give a total preference order over the alternatives. In this case, there must be at least as many contribution values as there are alternatives, since the total order is transitive.

- The preference relations over the alternatives do not give a complete order. In this case, the contribution relations cannot be defined from preference relations.

The two cases above suggest that a single contribution relation type is not expressive enough. Different softgoals will generate different combinations of preferences over alternatives, so that defining a single contribution relation type with any number n of contribution values is not recommended. Rather, if contribution relations are necessary, define preferences and produce a contribution relation type for each softgoal.

In the example used for illustration earlier (§7.1), the softgoal $\mathbf{sg}(\tilde{w}_1)$ generates the preference relation $\Pi_1 \succ \Pi_2$, where:

- Π_1 is a set which includes $\mathbf{t}(u_1)$ and $\mathbf{t}(u_2)$, along with the assumption $\mathbf{k}(\mathbf{t}(u_1) \wedge \mathbf{t}(u_2) \rightarrow \mathbf{g}(p)_1)$ and the responsibility assignments $\mathsf{R}_2\mathbf{t}(u_1)$ and $\mathsf{R}_2\mathbf{t}(u_2)$; in words, the set Π_1 describes the option which suggest to identify available ambulances using the dispatch software interface, which is responsible for executing both $\mathbf{t}(u_1)$, the task of showing the list of all available and allocated ambulances, and $\mathbf{t}(u_2)$, the task of updating that list whenever an ambulance is assigned to an incident;

- Π_2 is a set which includes the description of the option to identify available ambulances by having every control assistant, denoted Π_1, execute $\mathbf{t}(u_3)$, the task of manually keeping track of available ambulances; this set includes $\mathbf{t}(u_3)$, the assumption $\mathbf{k}(\mathbf{t}(u_3) \rightarrow \mathbf{g}(p_1))$, and the responsibility assignment $\mathsf{R}_1\mathbf{t}(u_3)$.

The contribution relations which correspond to the setup above need to have at least two contribution values. Let these be $+$ and $++$, where the former indicates weaker contribution than the latter. The contribution relations corresponding to the above are then $\mathsf{Contribute}(\Pi_1, ++, \mathbf{sg}(\tilde{w}_1))$ and $\mathsf{Contribute}(\Pi_2, +, \mathbf{sg}(\tilde{w}_1))$.

Suppose now that there is a third option which refines $\mathbf{g}(p_1)$, given by the set Π_3, and consists of having the dispatch interface list available and assigned ambulances, and having control assistants who update that list via the interface. There are now two possibilities with regards to generating contribution relations from this new setup:

- If (i) the relation $\succ$ is transitive, and (ii) $\Pi_1 \succ \Pi_3$ and $\Pi_3 \succ \Pi_1$, then also $\Pi_1 \succ \Pi_2$, and a three-valued contribution relation is needed. Let $+$, $++$, and $+++$ be the three contribution values, in increasing strength of positive contribution. The three contribution relations that capture this case are $\mathsf{Contribute}(\Pi_1, +++, \mathbf{sg}(\tilde{w}_1))$ and $\mathsf{Contribute}(\Pi_2, +, \mathbf{sg}(\tilde{w}_1))$ and $\mathsf{Contribute}(\Pi_3, ++, \mathbf{sg}(\tilde{w}_1))$.

- If the softgoal does not generate a total preference order over the three sets (e.g., there is no preference between Π_1 and Π_3), then a three-valued contribution relation is not appropriate, since the three sets cannot be compared using the strict preference relation.

The example above illustrates that contribution relations can be viewed as relations derived from preferences generated by softgoals, and that they exist only if the softgoal generates a complete preference order over the sets of requirements to which it applies.

7.4.2 Qualitative Relaxation Relation

A requirement is relaxed if it cannot be satisfied, because it is not feasible to do so. The process of relaxing a requirement consists of changing the requirements database so that the idealistic requirement no longer needs to be satisfied. The result is that other one or more requirements, this time feasible, need to be satisfied, instead of the idealistic requirement.

There are basically two ways to relax a requirement. One is to remove it from the requirements database, and add some requirements which appear to be feasible, and ought to be satisfied instead. The drawback of this approach is that the original requirement is lost from the requirements database. To trace back the reason for having the added requirements in the database, one has to look at a previous version of the requirements database, rather than in the current version of the requirements database. The second option is to relate the idealistic requirement to requirements which relax it, and thereby keep them all in the current version of the requirements database.

The relaxation relation between an idealistic requirement and the requirements which relax it is defined over inference and preference relations. Since the latter will be less desired than the former, the relaxation relation can also be called an approximation relation, in that satisfying the feasible requirements approximates, i.e., is good enough but not equal to, satisfying the idealistic requirement.

It is not precise enough to call this only a relaxation relation, since there is another way to relax requirements, which is discussed later for a more expressive Techne formalism. The appropriate name here is *qualitative relaxation relation*, because it can target only binary requirements, not those which place constraints on values of numerical variables.

Definition 7.4. There is a **qualitative relaxation relation** from the set $\{\Pi_1, \ldots, \Pi_n\}$ to the requirement $\phi \notin \Delta^{\rightarrow}$ if and only if:

1. $\forall 1 \leq i \leq n,\ \exists(\Pi_i, \phi) \in \mathsf{Arg}(\Delta)$, i.e., every Π_i is a premise in an argument for ϕ,

2. There is a total order using $\succ$, or a total preorder using $\succeq$ over $\{\Pi_1, \ldots, \Pi_n\}$, i.e., all premises of arguments for ϕ are ranked by the preference relations.

■

Recall that $\succeq$ gives a total preorder on $\{\Pi_1, \ldots, \Pi_n\}$ iff $\succ$ is transitive and total (i.e., compares every two members of that set). $\succ$ gives a total order on $\{\Pi_1, \ldots, \Pi_n\}$ iff it is irreflexive, transitive, and total.

The qualitative relaxation relation can be specialized in the obvious way, by adding constraints on the core modalities of the idealistic requirement, and/or

on the core modalities over the requirements in the argument premises. A "goal relaxation relation" could then be analogous to the goal refinement relation, in that the idealistic requirement must be a goal, and the requirements which approximate it must also be goals.

The goal $\mathbf{g}(p_2$: Fully automate ambulance availability identification) is idealistic since it is not feasible to automatically collect all information about ambulances (e.g., where they are, where they are going, what incident they are assigned to, and so on) which is needed to automate availability identification. To relax this goal, one starts by introducing at least two arguments, e.g., $(\Pi_1, \mathbf{g}(p_2))$ and $(\Pi_2, \mathbf{g}(p_2))$, where each premise describes one way to partly automate ambulance availability identification. The second step is to add preferences between the premises until there is a total order or a total preorder.

7.4.3 Qualitative Tradeoff Relation

Definition 7.5. There is a **qualitative tradeoff relation** between subsets Π_1 and Π_2 of Δ if and only if

1. $\Pi_1 \cup \Pi_2 \cup \Delta^{\rightarrow} \mathrel{|\!\sim_{\mathfrak{Z}}} \bot$, i.e., the two sets are inconsistent, and
2. at least one of the following cases holds, for $i, j \in \{1, 2\}$ and $i \neq j$:
   - $\Pi_i \succ \Pi_j$ and $\Pi_j \succ \Pi_i$, or
   - $\Pi_i \succ \Pi_j$ and $\Pi_j \succeq \Pi_i$.

In words, if there are contradictory preferences between the two sets, then there is an inconsistency. ■

The relation above involves a *tradeoff* because it is impossible to choose simultaneously both of the preferred sets of requirements. It is a *qualitative* tradeoff relation because it is a tradeoff over binary variables, as each requirement is either satisfied or not.

7.4.4 Labeled Preference Relations

Preference relations generated by softgoals and soft domain assumptions can be explicitly associated to the relevant softgoal or soft domain assumption. If a softgoal $\mathbf{sg}(\tilde{w})$ generates the preference $\Pi \succ \Psi$, labeling that preference relation gives $\succ_{\mathbf{sg}(\tilde{w})}$ and $\Pi \succ_{\mathbf{sg}(\tilde{w})} \Psi$, and allows the distinction between preferences generated by different softgoals and soft domain assumptions.

Adding a label specializes the unlabeled preference relation. It allows, e.g., the definition of database interface filters which apply only to a particular preference relation, which can be useful when there is a need to compute, e.g., the transitive closure over a preference relation, or establish if the preferences generated by a single softgoal give a total order.

7.5 Database Interface

All filters defined for **T2** apply in **T3**. Additional filters are defined below and apply to the preference relation and vague requirements; when $\succ$ is used without a label (cf., §7.4.4), it refers to all labeled and unlabeled preferences in the requirements database:

- All vague requirements which generate exactly n preference relations, for $\mathsf{x} \in \{\mathsf{sg}, \mathsf{sk}\}$:

$$\mathsf{APNo}(\mathsf{x}, n, \Delta) = \{\ \phi \in \Delta_{\mathsf{x}} \mid \nexists \phi \leftarrow X \in \Delta \text{ s.t.} |X| = n \text{ and } X = \{\Pi_i R \Pi_j \mid 1 \leq i, j \leq m,\ R \in \{\succ, \succeq, \approx\}\}\}. \quad (7.69)$$

- All qualitative tradeoffs:

$$\mathsf{AQTradeoff}(\Delta) = \{\Phi \mid \Phi = \{\Pi_i, \Pi_j\},\ \Pi_i \neq \Pi_j,\ \Pi_i, \Pi_j \in \wp(\Delta), \text{ and } \{\Pi_i \succ \Pi_j, \Pi_j \succ \Pi_i\} \subset \Delta \text{ or } \{\Pi_i \succ \Pi_j, \Pi_j \succeq \Pi_i\} \subset \Delta\}. \quad (7.70)$$

- All preferred requirements:

$$\mathsf{APreferred}(\Delta) = \{\phi \mid \phi \notin \Delta^{\rightarrow} \text{ and } \exists \Pi \in \wp(\Delta) \text{ s.t. } \phi \in \Pi \text{ and } \exists \Pi R \Psi \in \Delta,\ R \in \{\succ, \succeq\}\}. \quad (7.71)$$

- All dominated requirements:

$$\mathsf{ADominated}(\Delta) = \{\phi \mid \phi \notin \Delta^{\rightarrow} \text{ and } \exists \Pi \in \wp(\Delta) \text{ s.t. } \phi \in \Pi \text{ and } \exists \Psi R \Pi \in \Delta,\ R \in \{\succ, \succeq\}\}. \quad (7.72)$$

- All dominating requirements:

$$\mathsf{ADominating}(\Delta) = \mathsf{APreferred}(\Delta) \setminus \mathsf{ADominated}(\Delta). \quad (7.73)$$

- All criteria:

$$\mathsf{ACriteria}(\Delta) = \mathsf{TClosure}(\succ) \cup \mathsf{TClosure}(\succeq), \quad (7.74)$$

  where $\mathsf{TClosure}(\succ)$ is constructed as follows:

  1. For $\succ \subseteq \wp(\Delta) \times \wp(\Delta)$, let $\succ^0 \stackrel{def}{=} \succ$;
  2. For $i > 0$,

$$\succ^i \stackrel{def}{=} \succ^{i-1} \cup \{(\Pi_i, \Pi_k) \mid \exists \Pi_j \text{ s.t. } (\Pi_i, \Pi_k) \in \succ^{i-1} \text{ and } (\Pi_j, \Pi_k) \in \succ^{i-1}\}, \quad (7.75)$$

  3. Finally, $\mathsf{TClosure}(\succ) \stackrel{def}{=} \bigcup_{i \in \mathbb{N}} \succ^i$.

and $\mathsf{TClosure}(\succeq)$ is constructed in the analogous way, by replacing every occurence of $\succ$ with $\succeq$. Every criterion is a pair of requirements sets connected by a preference relation, whereby the pair is taken from the transitive closure of $\succ$ and $\succeq$.

Every filter above can be specialized for a particular softgoal or soft domain assumption. E.g., it may be interesting to find all qualitative tradeoffs which involve the softgoal $\mathbf{sg}(\tilde{w}$: Software is highly secure), which requires the following filter:

$$\begin{aligned}\mathsf{AQTradeoff}(x, \Delta) = \{\Phi \mid\ & \Phi = \{\Pi_i, \Pi_j\},\ \Pi_i \neq \Pi_j,\ \Pi_i, \Pi_j \in \wp(\Delta), \\ & \text{and } \{\Pi_i \succ_x \Pi_j, \Pi_j \succ_x \Pi_i\} \subset \Delta \\ & \text{or } \{\Pi_i \succ_x \Pi_j, \Pi_j \succeq_x \Pi_i\} \subset \Delta\}. \end{aligned} \tag{7.76}$$

where $x \in \Delta_{\mathbf{sg}} \cup \Delta_{\mathbf{sk}}$ is a softgoal or a soft domain assumption.

7.6 Discussion

Despite being built by extending **T2**, **T3** results in the requirements problem and solution concepts which strongly depart from those in both **T1** and **T2**. While every solution still has to satisfy the Consistency and Achievement conditions, it now also has to dominate – according to some decision rule – other solutions. Decision rules in turn are defined over preferences, while vague requirements act as sources of preferences. Adding these notions means that it is possible to explicitly give information necessary for decision-making which is considerably more sophisticated than the decision rule implicit in **T1** and **T2**, namely, choose any solution which satisfies the Consistency and Achievement conditions.

Allowing preferences as in **T4**, while having the basic core modalities, agents, and roles, shows that **T4** can express requirements and decision information which could not be captured in prior work, and formalizes a number of informally used notions:

- Preferences can be given also over alternative responsibility assignments, over alternative assignments of agents to roles, and over alternative commitments;
- The relationship of a softgoal to preference relations is that the former is a source of the latter;
- A softgoal is one specialization of a vague requirement, meaning that there can be other kinds of vague requirements, such as the soft domain assumption;
- As soon as a formalism has the preference relation, the contribution relation cannot be considered primitive, but a derived relation;
- Preferences can be in conflict, and such conflicts indicate tradeoffs;

- Responsibility, ability, commitment, and occupancy can be conditional on the satisfaction of requirements, and the other way around.

8 T4

T4 is built over **T3** by adding *optionality modalities*. They are three unary relations on requirements, used as follows:

- If a requirement must be satisfied in every solution of the requirements problem, then it is a mandatory – or equivalently, a necessary – requirement, and the Mandatory modality is used to state this.
- If it is desirable, but not necessary to satisfy a requirement, then the requirement carries the Preferred modality.
- If a requirement is in the premises of an argument *and* it is not a Mandatory requirement, then it carries the Inherited modality. Inheritance of optionality modalities is discussed further below.

The effects of adding the optionality modalities are as follows:

- It is not possible in **T3** to distinguish requirements which *must* be satisfied from requirements which are desired, but not necessary. E.g., it is necessary that reported incidents are handled, while it may be desirable, but not necessary that the exact location of every ambulance is always known to the dispatch center. This distinction is crucial for the definition of the solution concept. In **T3**, a solution was any set which satisfied the Consistency and Achievement conditions, the latter requiring that all top-level goals be satisfied: every top-level goal was thereby, by default, a mandatory goal. **T4** has the optionality modalities, and they result in a new Achievement condition for the solution concept.
- Because **T4** distinguishes necessary from non-necessary requirements, it has three solution concepts: *threshold solution*, *candidate solution*, and *solution*. A set of requirements must satisfy *all* mandatory requirement to be a threshold solution. A threshold solution is also minimal, in that it includes only the information necessary to satisfy all mandatory requirements. A candidate solution is made by adding new information to a threshold solution. A candidate solution which is chosen as the result of decision-making is called the solution.
- The preferred modality carries two pieces of information. Firstly, it says that it is not necessary to satisfy the preferred requirement: it may be satisfied by some solutions, but need not, and it does not need to be satisfied in every solution. Secondly, the preferred modality carries information about relative desirability, and thereby about preference. A requirement with a preferred modality can be understood as being involved in a special case of preference relation. Since there is no negation in Techne, $\phi \succ \neg\phi$ is

not an expression. Having the preferred modality on ϕ indicates that if two solutions differ *only* in that one satisfies ϕ, the other does not, then the former is preferred to the latter. It follows that a requirement with the preferred modality is a criterion, to be used alongside preferences in decision-making.

- Distinction between mandatory and preferred domain assumptions can be read as the distinction between definite and defeasible beliefs about the domain. A defeasible domain assumption is a domain assumption which carries the Preferred modality. Rather, the Preferred modality, when applied to a domain assumption can be informally read *in the absence of any information to the contrary*, i.e., $\mathsf{k}(\phi \rightarrow \psi)^{\mathsf{P}}$ reads "If ϕ, then, in the absence of any information to the contrary, ψ" as in Reiter's logic for default reasoning [25]. Another way to read it is *reasons to believe in the antecedent provide reasons to believe in the consequent*, as if it was a default rule in Simari & Loui's argumentation framework [28], namely, $\mathsf{k}(\phi \rightarrow \psi)^{\mathsf{P}}$ reads "reasons to believe in ϕ provide reasons to believe in ψ". It follows that a domain assumption which includes both an implication connective and carries the Preferred modality cannot be an axiom.

8.1 Illustration

Consider again the example in Figure 1. The goal $\mathbf{g}(q_1)$, that ambulances arrive at their incident locations should be satisfied in every solution, which makes it a mandatory goal, denoted $\mathbf{g}(q_1)^{\mathsf{M}}$.

The goals $\mathbf{g}(p_1)$ to $\mathbf{g}(p_5)$ refine $\mathbf{g}(q_1)$ in that figure. Every one of these goals inherits the mandatory modality from $\mathbf{g}(q_1)$, because all of them must be satisfied in order for $\mathbf{g}(q_1)$ to be satisfied. Their optionality modalities are, however, *not* mandatory, but inherited. The reason is that setting them all to mandatory would make any other refinement of $\mathbf{g}(q_1)$ irrelevant. To see why, suppose that there is another refinement of $\mathbf{g}(q_1)$, via goals $\mathbf{g}(p_6)$ to $\mathbf{g}(p_{10})$, i.e., there is a domain assumption $\mathsf{k}(\bigwedge_{i=6}^{10} \mathbf{g}(p_i) \rightarrow \mathbf{g}(q_1))$. If it is only known that $\mathbf{g}(q_1)^{\mathsf{M}}$, i.e., that $\mathbf{g}(q_1)$ is mandatory, then any of the following can be done:

- If no optionality modality is assigned to the goals in the two refinements, making any one goal mandatory in one of the two refinements results in the decision to reject the other refinement. E.g., if $\mathbf{g}(p_9)$ is made mandatory, then every solution must satisfy $\mathbf{g}(p_9)$, and the refinement $\mathsf{k}(\bigwedge_{i=1}^{5} \mathbf{g}(p_i) \rightarrow \mathbf{g}(q_1))$ becomes irrelevant.

- If any of the goals in either refinement is made optional or preferred, then this is an error in the use of **T4**. Because all goals in a refinement must be satisfied in order for the refined goal to be satisfied, if the latter is mandatory, then the former cannot be unnecessary.

- The role of the inherited optionality modality is to allow a requirement, say ϕ, to have an undecided status, yet a status which – when decided – is

settled based on the optionality modality of all requirements $\psi_1, \ldots, \psi_n$ such that $\phi \in \Pi_i$ and $(\Pi_i, \psi_i) \in \mathsf{Arg}(\Delta)$. In words, if ϕ is among the premises of an argument for a requirement ψ_i, then the optionality modality of ϕ depends on the optionality modality of ψ_i, unless ϕ is already has the Mandatory modality. Returning to $\mathbf{g}(q_1)^{\mathsf{M}}$, the optionality modalities of requirements in all arguments which conclude $\mathbf{g}(q_1)^{\mathsf{M}}$ should have the inherited optionality modality, including thereby all requirements in both refinements of that goal.

The Inherited modality can be seen as postponing the decision about the optionality modality of a requirement. If viewed so, the moment at which the decision is made is the moment at which a solution is chosen for the requirements problem. In the example above, if the chosen solution includes the refinement $\mathsf{k}(\bigwedge_{i=6}^{10} \mathbf{g}(p_i) \rightarrow \mathbf{g}(q_1))$, then every goal $\mathbf{g}(p_6), \ldots, \mathbf{g}(p_{10})$ which has the modality Inherited can be read as having the modality Mandatory: being part of the chosen solution, and since they refine a mandatory requirement, these requirements can be viewed as mandatory too. The Inherited modality thus serves to avoid premature decision-making about how a requirement should be operationalized.

8.2 Formalization

Semantic Domain. The semantic domain of **T4** includes vague and clear natural language propositions, agents, and roles. **T4** carries over all relations from **T3**, and adds three unary relations for the three optionality modalities: Mandatory, Preferred, and Inherited. The optionality modalities are informally defined as follows:

- A clear natural language proposition has the Mandatory modality if and only if every solution to the requirements problem must satisfy the conditions stated by that proposition.
- A clear natural language proposition has the Preferred modality if and only if it is not necessary that every or any solution to the requirements problem satisfies it, but it is desirable that it is satisfied by the chosen solution.
- A clear natural language proposition has the Inherited modality if and only if it serves as a premise in at least one inference relation *and* it does not have the Mandatory modality.

Syntax. **T4** maintains the distinction between modeling and decision-making parts of the language. Expressions in the modeling part obtain symbols for optionality modalities.

The language of **T4** is the union $\mathcal{L}_{\mathbf{4}} = \mathcal{L}_{\mathbf{4}}^{M} \cup \mathcal{L}_{\mathbf{4}}^{D}$ of expressions where every $\phi \in \mathcal{L}_{\mathbf{4}}^{M}$ and every $\tilde{\phi} \in \mathcal{L}_{\mathbf{4}}^{D}$ satisfies the following BNF specification:

$$a ::= \mathbf{g}(p) \mid \mathbf{k}(p) \mid \mathbf{t}(p) \tag{8.77}$$
$$b ::= \mathsf{R}a \mid \mathsf{A}a \mid \mathsf{cA}a \tag{8.78}$$
$$c ::= a \mid b \mid \mathsf{O}(\mathsf{R}, \mathsf{A}) \tag{8.79}$$
$$d ::= c^{\mathsf{M}} \mid c^{\mathsf{P}} \mid c^{\mathsf{H}} \tag{8.80}$$
$$e ::= (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow c \mid (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow \bot \tag{8.81}$$
$$f ::= \mathbf{k}(e)^{\mathsf{M}} \mid \mathbf{k}(e)^{\mathsf{P}} \mid \mathbf{k}(e)^{\mathsf{H}} \tag{8.82}$$
$$\phi ::= d \mid f \tag{8.83}$$
$$g ::= \{d_1, \ldots, d_{n\geq 1}\} \tag{8.84}$$
$$h ::= g \succ g \mid g \approx g \tag{8.85}$$
$$q ::= \{h_1, \ldots, h_{n\geq 0}\} \tag{8.86}$$
$$\tilde{\phi} ::= q \mid \mathbf{sg}(\tilde{p}) \leftarrow q \mid \mathbf{sk}(\tilde{p}) \leftarrow q \mid q \rightarrow \bot \tag{8.87}$$

The first seven rules revisit the grammar of $\mathcal{L}_{\mathbf{3}}^{M}$, adding symbols for optionality modalities. Goals, domain assumptions, and tasks, as well as the responsibility, ability, occupancy, and commitment can carry an optionality modality. The syntax reflects the idea that every requirement in a model should carry one of the three optionality modalities.

The language $\mathcal{L}_{\mathbf{4}}$ has a partition analogous to $\mathcal{L}_{\mathbf{3}}$, that is, $\mathcal{L}_{\mathbf{4}} = \mathcal{L}_{\mathbf{4}}^{M} \cup \mathcal{L}_{\mathbf{4}}^{D}$, with $\mathcal{L}_{\mathbf{4}}^{M} \cap \mathcal{L}_{\mathbf{4}}^{D} = \emptyset$.

Optionality modalities remain in the modeling part only. Preference relations and vague requirements cannot carry optionality modalities. There is no reason other than simplicity to keep optionality modalities outside $\mathcal{L}_{\mathbf{4}}^{D}$.

Semantic Mapping. Informal discussions up to this point suggest how symbols and expressions map to members of the semantic domain.

Consequence Relation. Optionality modalities have no influence on the consequence relation, so that the consequence relation is defined as in **T3**.

Definition 8.1. For $\Pi \subseteq_{\tau} \mathcal{L}_{\mathbf{4}}$ and $\phi \in \mathcal{L}_{\mathbf{4}}$, the **consequence relation** $\mid\!\sim_{\mathbf{4}}$ is such that:

- $\Pi \mid\!\sim_{\mathbf{4}} \phi$ if $\phi \in \Pi$, or
- $\Pi \mid\!\sim_{\mathbf{4}} x$ if $\forall 1 \leq n$, $\Pi \mid\!\sim_{\mathbf{4}} \phi_i$ and $\mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow x) \in \Pi$,
- $\mid\!\sim_{\mathbf{5}}$ satisfies the *preference conflict* axioms:

$$\forall \Pi_i, \Pi_j \subseteq \Pi, \ \ \{\Pi_i \succ \Pi_j, \Pi_j \succ \Pi_i\} \rightarrow \bot, \tag{8.88}$$
$$\forall \Pi_i, \Pi_j \subseteq \Pi, \ \ \{\Pi_i \succeq \Pi_j, \Pi_j \succ \Pi_i\} \rightarrow \bot. \tag{8.89}$$

The preference conflict axioms indicate that contradictory preferences must produce inconsistency. ■

Remarks on soundness and completeness made for $\mid\!\sim_{\mathbf{1}}$ and $\mid\!\sim_{\mathbf{2}}$ apply for $\mid\!\sim_{\mathbf{4}}$.

8.3 Problem & Solution Concepts

Optionality modalities add information which is useful in two respects in the definition of the problem and solution concepts:

- The Mandatory modality results in the distinction between three solution concepts, namely *threshold solution* as that which satisfies all Mandatory requirements only, *candidate solution*, as a threshold solution expanded to satisfy Preferred requirements, and the *solution* concept as the candidate solution which was chosen among various candidate solutions.
- The Preferred modality adds new information usefeul in the definition of decision rules. Recall that a decision rule in **T3** served to rank solutions on the basis of preference relations. In **T4**, decision rules can be defined to use both preference relations and the Preferred unary relations on requirements in order to rank solutions.

The requirements problem in **T4** is stated in the same way as in **T3**.

Definition 8.2. Given a set of goals, domain assumptions, and agents, find the solution which ranks highest according to the chosen decision rule. ■

The decision rule is again a macro which returns a ranking of the solutions, given a set of solutions, preference relations and optionality unary relations over requirements. E.g., a macro can be defined to rank solutions from the solution which satisfies most preferred requirements in preference relations and most requirements with Preferred modality.

The axioms set definition needs to be revised so that the set of axioms includes only requirements which have both the implication connective and the Mandatory modality.

Definition 8.3. The set of **axioms** in a requirements database $\Delta \subseteq \mathcal{L}_{\mathbf{4}}$ is the set:

$$\begin{aligned}\Delta^{\rightarrow} \stackrel{def}{=} \{\phi \mid \phi \text{ has the implication connective and } \} \\ \phi \text{ has the Mandatory optionality modality}\}. \end{aligned} \tag{8.90}$$

■

The Select function also needs to be revised, so that subsets of a requirements database can be extracted which carry a particular optionality modality.

Definition 8.4. The **select function** in **T4** is defined as follows:

$$\begin{aligned}\mathsf{Select}(\mathsf{x},\mathsf{y},\Pi) \stackrel{def}{=} \{\phi \mid \phi \text{ has the core modality } \mathsf{x} \\ \text{and has the optionality modality } \mathsf{y}, \text{ or } \phi \in \Pi^{\rightarrow}\}. \end{aligned} \tag{8.91}$$

The part "or $\phi \in \Pi^{\rightarrow}$" ensures that the output of the select function always includes all axioms. A modality can be a core modality or an agency modality. ■

To simplify notation, the following abbreviations are used:

$$\mathsf{Select}(\mathbf{x}, \mathbf{y}, \Pi) \equiv \Pi^{\mathbf{y}}_{\mathbf{x}}, \tag{8.92}$$

$$\Pi_{\mathbf{x}} \equiv \bigcup_{\mathbf{y} \in \{\mathsf{M},\mathsf{P},\mathsf{H}\}} \Pi^{\mathbf{y}}_{\mathbf{x}}, \tag{8.93}$$

$$\Pi^{\mathbf{y}} \equiv \bigcup_{\mathbf{x} \in \{\mathbf{g},\mathbf{k},\mathbf{t}\}} \Pi^{\mathbf{y}}_{\mathbf{x}}. \tag{8.94}$$

In words, $\Pi_{\mathbf{x}}$ is the subset of Π which have the core modality $\mathbf{x}$ and have any optionality modality. $\Pi^{\mathbf{y}}$ is the subset of Π which has the optionality modality $\mathbf{y}$ and has any core modality.

Definition 8.5. A **threshold solution** to the requirements problem given by a requirements database $\Delta \subseteq \mathcal{L}_4$ is a set $S \subseteq_\tau \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ of domain assumptions and tasks, which satisfies the following four properties:

1. *Consistency:* $S \not\vdash_4 \bot$, i.e., all requirements in a threshold solution must be consistent;
2. *Threshold achievement:* $\forall \phi \in \Delta^{\mathsf{M}}_{\mathbf{g}}$, $\exists \Pi \in \mathsf{Op}(\phi)$ s.t. $\Pi \subseteq S$, i.e., a threshold solution must include at least one operationalization of every Mandatory goal;
3. *Conformity:* $\forall \phi \in \Delta^{\mathsf{M}}_{\mathbf{k}} \cup \Delta^{\mathsf{M}}_{\mathbf{t}}$, $S \vdash_4 \phi$, i.e., a threshold solution must satisfy all Mandatory domain assumptions and all Mandatory tasks;
4. *Minimality:* $\nexists S' \subset S$ s.t. S' satisfies the Consistency, Threshold achievement, and Conformity conditions, i.e., a threshold solution must include only the requirements necessary to satisfy the three conditions above.

■

The Threshold achievement condition requires that all Mandatory goals be satisfied, while the Conformity condition requires asks that all other Mandatory requirements are satisfied too. These two conditions correspond to the Achievement condition in **T3** and simpler Techne formalisms. The Minimality condition is necessary in **T4** in order to distinguish threshold solutions from the candidate solution concept.

Definition 8.6. A **candidate solution** to the requirements problem given by a requirements database $\Delta \subseteq \mathcal{L}_4$ is a set $S \subseteq_\tau \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ of domain assumptions and tasks, which satisfies the following four properties:

1. *Threshold inclusion*: $\exists S' \subseteq_\tau S$, where S' is a threshold solution to the requirements problem in Δ, i.e., S is equivalent or superset of a threshold solution;

2. *Expansion:* there is a potentially empty $\Pi \subseteq \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ such that $S = S' \cup \Pi$, i.e., there is a potentially empty set of domain assumptions and tasks which are not Mandatory and which are included in S.

A candidate solution is either equal to a threshold solution (when Π is empty) or is a threshold solution which was expanded by adding non-Mandatory tasks and domain assumptions to it. ■

Observe the following:

- The aim of decision-making is to rank candidate solutions and thereby recommend the highest-ranking one as the solution to the requirements problem. If a threshold solution cannot receive any more tasks or domain assumptions and remain consistent, then $\Pi = \emptyset$ in Definition 8.6, and the threshold solution can be, as-is, called a candidate solution. Definiton 8.6 allows $\Pi = \emptyset$ in order to allow a threshold solution to appear among candidates, and thereby allow one to choose a candidate solution which fails to have any but the minimal set of tasks and domain assumptions which satisfy the Consistency, Threshold achievement, Conformity, and Minimality condition.

- There can be many candidate solutions where $\Pi \neq \emptyset$. Definition 8.6 effectively allows as a candidate solution *any* expansion of a threshold solution, as long as the expanded candidate solution is consistent. That definition thereby *does not* require every candidate solution to be *maximally consistent*. If one wishes to rank only maximally consistent candidates, then this should be done via a decision rule, as it is straightforward to define a decision rule which will only allow maximally consistent candidates. Asking that every candidate is maximally consistent would be to have too many constraints in the candidate solution definition, and thereby miss potentially relevant solutions which are not maximally consistent.

Preferences and Preferred requirements remain outside the solution concepts. As in **T3**, they are used for the definition of decision rules.

8.4 Derived Relations

T4 keeps all relations defined in **T1**–**T3** and changes the definitions of the inference and conflict relations.

8.4.1 Strict & Defeasible Inference

Definition 5.6 of the inference relation required that the domain assumption which relates the premises and the conclusion be an axiom. **T4** cannot keep the inference relation as-is, since it allows domain assumptions to be axioms, when they are Mandatory, or defeasible, when they are Preferred. There are two inference relations now, as follows.

Definition 8.7. A requirement $\phi \in \Delta$ stands in the **inference relation** with the requirements $\{\psi_1, \ldots, \psi_n\} \subseteq \Delta$, $n \geq 1$, if and only if:

1. $\mathbf{k}((\bigwedge_{i=1}^{n} \psi_i) \rightarrow \phi)^{\mathsf{M}} \in \Delta^{\rightarrow}$;
2. $\nexists \Pi \subseteq_{\tau} \Delta$ s.t. $\Pi \subset_{\tau} (\{\psi_1, \ldots, \psi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \mathrel{\mid\!\!\sim_{\mathfrak{1}}} \phi$;
3. $\nexists \gamma \in \Delta^{\rightarrow}$ s.t. $\{\gamma\} \cup \{\bigwedge_{i=1}^{n} \psi_i\} \mathrel{\mid\!\!\sim_{\mathfrak{1}}} \bot$.

■

The only difference between inference in **T1**–**T3** and the strict inference relation above is that the domain assumption connecting premises and the conclusion is Mandatory. In the defeasible inference, that domain assumption is instead Preferred, and thereby not an axiom.

Definition 8.8. A requirement $\phi \in \Delta$ stands in the **inference relation** with the requirements $\{\psi_1, \ldots, \psi_n\} \subseteq \Delta$, $n \geq 1$, if and only if:

1. $\mathbf{k}((\bigwedge_{i=1}^{n} \psi_i) \rightarrow \phi)^{\mathsf{P}} \in \Delta$;
2. $\nexists \Pi \subseteq_{\tau} \Delta$ s.t. $\Pi \subset_{\tau} (\{\psi_1, \ldots, \psi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \mathrel{\mid\!\!\sim_{\mathfrak{1}}} \phi$.
3. $\nexists \gamma \in \Delta^{\rightarrow}$ s.t. $\{\gamma\} \cup \{\bigwedge_{i=1}^{n} \psi_i\} \mathrel{\mid\!\!\sim_{\mathfrak{1}}} \bot$.

■

Because $\mathbf{k}((\bigwedge_{i=1}^{n} \psi_i) \rightarrow \phi)^{\mathsf{P}}$ is Preferred, it is only required in the third condition that no axiom contradicts it.

A convenient way to view the distinction above in light of **T1**–**T3**, is that the inference relation from those formalisms was not specialized, but that the defeasible inference relation was added alongside the strict inference relation. This fits the fact that both the strict and defeasible inference relation can each be specialized in the analogous way to the inference relation, e.g., onto, respectively, strict and defeasible goal refinement, strict and defeasible task decomposition, and so on.

8.4.2 Strict & Defeasible Conflict

The conflict relation that was relevant in **T1**–**T3** becomes the strict conflict relation in **T4**. The defeasible conflict relation differs from strict conflict only by having a defeasible domain assumption which connects the premises to the conclusion.

Definition 8.9. Requirements $\{\phi_1, \ldots, \phi_n\} \subseteq \Delta$ stand in the **strict conflict**, for $n \geq 2$, if and only if:

1. $\exists \mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow \bot)^{\mathsf{M}} \in \Delta^{\rightarrow}$;
2. $\nexists \Pi \subseteq \Delta$ s.t. $\Pi \subset (\{\phi_1, \ldots, \phi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \mathrel{\mid\!\!\sim_{\mathfrak{1}}} \phi$.

■

Definition 8.10. Requirements $\{\phi_1, \ldots, \phi_n\} \subseteq \Delta$ stand in the **defeasible conflict**, for $n \geq 2$, if and only if:

1. $\exists \mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow \bot)^{\mathsf{P}} \in \Delta^{\rightarrow}$;

2. $\nexists \Pi \subseteq \Delta$ s.t. $\Pi \subset (\{\phi_1, \ldots, \phi_n\} \cup \Delta^{\rightarrow})$ and $\Pi \mathrel{\not\sim_1} \phi$.

■

The change from the conflict relation to the strict and defeasible conflict relations is analogous to the change from the inference relation to the strict and defeasible relations.

Each of the strict and defeasible conflict relations in **T4** can be specialized in the exact same way as the conflict relation in **T1**–**T3**. E.g., defeasible conflict is specialized onto defeasible Type-A, defeasible Type-B , and defeasible Type-C conflicts. The definitions are not reproduced here as they are obvious from earlier discussions (cf., §5.4.2).

8.5 Database Interface

All filters defined for **T1**–**T3** apply in **T4**. There are no filters specific to **T4**.

9 T5

T5 is built over **T4** by making three major additions:

- The Quality constraint core modality is added. It cannot be associated to natural language propositions, but only to mathematical expressions over numerical variables. Its purpose is to identify desirable values of numerical variables. E.g., if the variable t equals the average time for an ambulance to reach an incident after the ambulance is mobilized to that incident, then $\mathbf{q}(t \leq 12\text{min})$ is a quality constraint which indicates that it is desirable for t to take a value up to 12 minutes.

- Assumptions can be made about, among others, arithmetic relations between numerical variables. In **T1**–**T4**, there were simple domain assumptions, namely, propositions associated to the modality $\mathbf{k}$, and complex domain assumptions, which include the implication connective. Now, e.g., an expression $v_1 + v_2 \leq v_3$, where the three variables are rational, can be a domain assumption, namely, $\mathbf{k}(v_1 + v_2 \leq v_3)^{\mathsf{P}}$ is an allowed expression in the language of **T5**.

- Mathematical relations over rational variables can carry the Task modality. This is allowed in order to capture tasks which consist of assigning a value to a particular variable.

- The new expressions above correspond, in the semantic domain, to relations between numerical variables. One such relation is quantitative refinement. E.g., the domain assumption $\mathbf{k}(x_1^2 + x_2 = x_3)$ gives the value of x_3 as a function of values of x_1 and x_2, and this is interpreted in **T4** as x_3 being refined by x_1 and x_2 according to the function given in the domain assumption.

- Relaxation of idealistic requirements was, in **T3**, only possible via qualitative relaxation. Allowing rational variables and quality constraints results now in the possibility to quantify, if needed, the level of satisfaction of a softgoal or of a soft domain assumption, through either probability functions or fuzzy membership functions. A softgoal for instance can be treated as having a level of satisfaction described by a continuous function of a numerical variable: e.g., $f(t)$ may be defined to return the satisfaction level for a given time t for an ambulance to reach its designated incident location. If one prefers to relax the softgoal via a probability function, it is then not the level of satisfaction achieved by the value of a variable that is of interest, but the frequency at which that variable obtains some value in a given range.

9.1 Illustration

In LAS, and in relation to the response to emergency calls and the mobilization of ambulances, the time that these activities take is a crucial part of quality of service. One can use the following variables, where c is a unique identifier of a call and e of a unique incident:

$t_{1,c}$: Time the caller waits for a control assistant;
$t_{2,c}$: Time to identify incident location;
$t_{3,c}$: Time to fill out incident report;
$t_{4,e}$: Time to mobilize an ambulance;
$t_{5,e}$: Time for the mobilized ambulance to arrive and confirm arrival at incident location.

A government standard may not be so specific as to define bounds on every one of these variables. Instead, it may impose a maximal time over a subset of activities that need to be executed between the reception of an emergency call and the confirmation of the arrival of an ambulance to the incident location; for example:

> $\mathbf{q}(t_{6,c} \leq 3\text{min}$: $t_{6,c}$ is the duration between the switching of the call c to the dispatch center to the mobilization of the ambulance to the incident location;)

where it is clear that the value of $t_{6,c}$ depends on $t_{1,c}$, $t_{2,c}$, $t_{3,c}$ and $t_{4,e}$. If it is assumed that $t_{6,c} = t_{1,c} + t_{2,c} + t_{3,c} + t_{4,e}$, then add this to the requirements database as a domain assumption over quantitative variables, $\mathbf{k}(t_{6,c} = t_{1,c} + t_{2,c} + t_{3,c} + t_{4,e})$. The domain assumption says that there is a **quantitative refinement** relation between variables, and it specifies the functional relation between the refined $t_{6,c}$ and the variables refining it. In contrast to the refinement relation, quantitative refinement is not over requirements, but variables in requirements.

While it may be useful to set a precise bound on the value of an aggregate variable, as $t_{6,c}$ above, it may be more interesting to set bounds relative to the values of other variables. E.g., if one wants $t_{2,c}$ to be at most 110% of its average value over the past three months, then this can be written

$\mathbf{q}(t_{2,c} \leq (1.1/3)(v_{1,m-3} + v_{1,m-2} + v_{1,m-1}))$;
$\mathbf{k}(v_{1,m} = n(m)^{-1} \sum_{i=1}^{n(m)} \sum_{c=1}^{c(i)} t_{2,c})$, where m is the month identifier, $n(m)$ the number of days in m, c is the call identifier on a given day, $c(i)$ is the total number of calls received on day i of month m;

where the domain assumption is a quantitative refinement of $v_{1,m}$, the average time, over a month, that it takes to identify the location.

The quality constraints and quantitative refinements need to be related to goals, tasks and domain assumptions in order to determine if the former are satisfied. If there is a running system, the values of $t_{2,c}$ are recorded, and it is straightforward to check if the constraint $t_{2,c} \leq (1.1/3)(v_{1,m-3} + v_{1,m-2} + v_{1,m-1})$ is satisfied. Before the system is in operation, values of $t_{2,c}$ can be simulated by assuming that $t_{2,c}$ is a random variable which has some probability distribution. E.g., $\mathbf{k}(t_{2,c} \backsim \mathcal{N}(60sec, 45sec^2))$ if it is assumed that $t_{2,c}$ follows a normal distribution with mean 60sec and variance 45sec^2 when the task $\mathbf{t}(u_1)$ is satisfied, which can be modeled by a refinement $\mathbf{k}(\mathbf{t}(u_1) \rightarrow \mathbf{k}(t_{2,c} \backsim \mathcal{N}(60sec, 45sec^2)))$. This assumption may be based on data from a pilot study, from expert opinion, or from data on systems which also satisfy $\mathbf{t}(u_1)$ and are already in operation.

An important consequence of allowing numerical variables is that it becomes possible to perform quantitative relaxation. The upper bound on $t_{2,c}$ in $\mathbf{q}(t_{2,c} \leq (1.1/3)(v_{1,m-3} + v_{1,m-2} + v_{1,m-1}))$ may still be too idealistic, as callers may provide information of very different quality about the incident location. The requirement can be relaxed in two alternative ways, by probabilistic or fuzzy relaxation.

Probabilistic relaxation of a quality constraint is done in two steps. Firstly, the variable constrained in $\mathbf{q}$ is redefined as a random variable, and an assumption is made on the probability distribution of that random variable. Above, $\mathbf{k}(t_{2,c} \backsim \mathcal{N}(60sec, 45sec^2))$ makes of $t_{2,c}$ a random variable which follows a normal distribution. Secondly, the quality constraint to be relaxed is removed from the requirements database, and a new quality constraint is added. The new constraint specifies a bound not on the value of the now random variable, but on the probability that its value is in some range. Since $t_{2,c}$ was made into a random variable in the first step, $\mathbf{q}(t_{2,c} \leq (1.1/3)(v_{1,m-3} + v_{1,m-2} + v_{1,m-1}))$ is now replaced with $\mathbf{q}(P(t_{2,c} \leq (1.1/3)(v_{1,m-3} + v_{1,m-2} + v_{1,m-1})) \geq 0.90)$, that is, it is now required that the minimal probability should be 0.90 for $t_{2,c}$ to be at most 110% of its three-month average. Letier & van Lamsweerde [20] suggested probabilistic relaxation, and argued that this approach to relaxation is more appropriate than fuzzy relaxation when there is quantitative data from which to estimate probability distributions, or when experts are confident enough to make assumptions about probability distributions.

Random variables and quality constraints thereon play an important role in decision-making, as they are used to set desired probability levels over random variables. In more informal terms, this means that one can write quality constraints and domain assumptions which reflect, respectively, the desired level of confidence that the system behaves in some way, and the assumed level of

confidence that may already be the case.

To illustrate how confidence can be expressed and used in domain assumptions, consider the goal $\mathbf{g}(p_3)$ in Figure 1. It has two operationalizations: one ($\mathbf{t}(u_8)$) involves the use of dispatch software to keep track of ambulance assignments, the other ($\mathbf{t}(u_9)$) is manual and paper-based. The two operationalizations results in different probability of erroneous ambulance assignments, and this is written, in **T5**, $\mathbf{k}(\mathbf{t}(u_8) \rightarrow \mathbf{q}(P(v_{2,i} \geq 5) \leq 0.10))$ for $\mathbf{t}(u_8)$ and $\mathbf{k}(\mathbf{t}(u_9) \rightarrow \mathbf{q}(P(v_{2,i} \geq 5) \geq 0.70))$ for $\mathbf{t}(u_9)$, where $v_{2,i}$ is the number of erroneous ambulance assignments on day i. This introduces a criterion for the comparison of $\mathbf{t}(u_8)$ and $\mathbf{t}(u_9)$, and the two domain assumptions reflect the belief that $\mathbf{t}(u_8)$ is better *on this criterion (and independently from other criteria)* than $\mathbf{t}(u_9)$.

Fuzzy relaxation of a quality constraint involves three steps:

1. Remove the quality constraint,
2. Define a fuzzy membership function over the variable from the removed quality constraint,
3. Define a softgoal over the variable from the removed quality constraint.

For illustration, consider again $\mathbf{q}(t_{2,c} \leq (1.1/3)(v_{1,m-3} + v_{1,m-2} + v_{1,m-1}))$ and apply fuzzy relaxation. Start by removing this quality constraint from the requirements database. A fuzzy membership function over $t_{2,c}$, denote it $\mu(t_{2,c})$, depends on how stakeholders evaluate in terms of desirability the various values of $t_{2,c}$. E.g., one could use $\mu(t_{2,c}) = e^{-t_{2,c}}$, so that the higher the value of $t_{2,c}$, the lower $\mu(t_{2,c})$ is, which reflects the idea that the more time it takes to identify an incident location, the more the stakeholders are dissatisfied, whereby their satisfaction increases as $t_{2,c}$ approaches 0. If one adopts μ as defined, the quality constraint on $t_{2,c}$ is removed and the level of satisfaction is quantified as a function of $t_{2,c}$.

To finish with the fuzzy relaxation of the quality constraint on $t_{2,c}$, a softgoal needs to be added to the requirements database over values of $t_{2,c}$. When used in fuzzy relaxation, a softgoal is defined over a known variable: in this example, it is reasonable to prefer lower over higher values of $t_{2,c}$, and there is consequently the following softgoal.

$\mathbf{sg}(\tilde{p}$: Low $t_{2,c}) \stackrel{def}{=}$
(Let $x_1 \stackrel{def}{=} \textsc{val}(S_1, t_{2,c})$ and $x_2 \stackrel{def}{=} \textsc{val}(S_2, t_{2,c})$,

- if $\mu(x_1) \geq \mu(x_2)$, then add $\{\mathbf{k}(t_{2,c} = x_1)\} \succeq \{\mathbf{k}(t_{2,c} = x_2)\}$ to Δ,
- if $\mu(x_1) > \mu(x_2)$, then add $\{\mathbf{k}(t_{2,c} = x_1)\} \succ \{\mathbf{k}(t_{2,c} = x_2)\}$ to Δ,
- if $\mu(x_1) = \mu(x_2)$, then add $\{\mathbf{k}(t_{2,c} = x_1)\} \approx \{\mathbf{k}(t_{2,c} = x_2)\}$ to Δ.)

Above, $\textsc{val}(S_1, t_{2,c})$ returns the value of $t_{2,c}$ in S_1. Although the formulation above seems very different from saying "Low $t_{2,c}$", this is precisely what it does. Note that $\mathbf{k}(t_{2,c} = x_1) \succ \mathbf{k}(t_{2,c} = x_2)$ says that satisfying $\mathbf{k}(t_{2,c} = x_1)$ is strictly

more desirable than satisfying $\mathsf{k}(t_{2,c} = x_2)$. The softgoal gives a macro which generates preference relations over domain assumptions. For any two candidate solutions S_1 and S_2, the macro compares the values that $t_{2,c}$ has in each of those two candidate solutions, and each of these values is recorded in a domain assumption of the form $t_{2,c} = const$ where *const* is a constant.

Depending on the comparison between the constants, the macro adds a preference relation to Δ. The macro ensures that if one compares two candidate solutions, S_1 and S_2, and $t_{2,c}$ obtains the value x_1 in S_1 and the value x_2 in S_2, then one will prefer *over this criterion (independently of other criteria)* the candidate in which $t_{2,s}$ obtains the lower value. In other words, the macro conveys the idea that, whenever two values of $t_{2,c}$ are given, the lower is preferred.

Fuzzy relaxation of a quality constraint over a variable v thus works by (i) removing the quality constraint, (ii) adding a fuzzy membership function $\mu(v)$ on v, (iii) interpreting $\mu(v)$ as the level of satisfaction with the value v, and (iv) adding a softgoal macro which generates preference relations that reflect the shape of μ.

9.2 Formalization

Semantic Domain. **T5**'s semantic domain includes clear and vague natural language propositions, rational numbers, agents, and roles. **T5** keeps all relations from **T4**, and adds various primitive relations between rational numbers. These relations have their standard definitions, and their list is given when syntax is defined below.

Relations over rational numbers are either instances of the Quality constraint concept, or of the Domain assumption concept. Expressions which state these relations over variables which take rational numbers intensionally define sets of rational numbers, i.e., those rational numbers which satisfy the relations stated in the expression.

Syntax. The language $\mathcal{L}_{\mathbf{5}}$ distinguishes the modeling $\mathcal{L}_{\mathbf{5}}^{M}$ and the decision-making $\mathcal{L}_{\mathbf{5}}^{D}$ parts. Of the two, it is the modeling part that obtains symbols for the representation of variables ranging over rational numbers, and symbols for the relations between these variables.

The language $\mathcal{L}_{\mathbf{5}}$ of **T5** is the union $\mathcal{L}_{\mathbf{5}} = \mathcal{L}_{\mathbf{5}}^{M} \cup \mathcal{L}_{\mathbf{5}}^{D}$ where every $\phi \in \mathcal{L}_{\mathbf{5}}^{M}$ and every $\tilde{\phi} \in \mathcal{L}_{\mathbf{5}}^{D}$ satisfies the following BNF specification in Equations 9.95–9.109:

$$w ::= \mathbf{g}(p) \mid \mathbf{k}(p) \mid \mathbf{t}(p) \tag{9.95}$$
$$x ::= n \mid v \mid x + x \mid x - x \mid x \cdot x \mid x/x \mid x^x \tag{9.96}$$
$$y ::= x > x \mid x < x \mid x = x \mid x \geq x \mid x \leq x \mid x \neq x \mid x \backsim \mathit{pdf} \tag{9.97}$$
$$z ::= \mathbf{q}(y) \mid \mathbf{k}(y) \mid \mathbf{t}(y) \tag{9.98}$$
$$a ::= w \mid z \tag{9.99}$$
$$b ::= \mathsf{R}a \mid \mathsf{A}a \mid \mathsf{cA}a \tag{9.100}$$
$$c ::= a \mid b \mid \mathsf{O}(\mathsf{R}, \mathsf{A}) \tag{9.101}$$
$$d ::= c^{\mathsf{M}} \mid c^{\mathsf{P}} \mid c^{\mathsf{H}} \tag{9.102}$$
$$e ::= (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow c \mid (\bigwedge_{i=1}^{n\geq 1} c_i) \rightarrow \bot \tag{9.103}$$
$$f ::= \mathbf{k}(e)^{\mathsf{M}} \mid \mathbf{k}(e)^{\mathsf{P}} \mid \mathbf{k}(e)^{\mathsf{H}} \tag{9.104}$$
$$\phi ::= d \mid f \tag{9.105}$$
$$g ::= \{d_1, \ldots, d_{n\geq 1}\} \tag{9.106}$$
$$h ::= g \succ g \mid g \approx g \tag{9.107}$$
$$q ::= \{h_1, \ldots, h_{n\geq 0}\} \tag{9.108}$$
$$\tilde{\phi} ::= q \mid \mathbf{sg}(\tilde{p}) \leftarrow q \mid \mathbf{sk}(\tilde{p}) \leftarrow q \mid q \rightarrow \bot \tag{9.109}$$

The main change in syntax from **T4** happens in the first four rules in the BNF specification. With the first rule, which generates ws, natural language propositions remain labeled with the three core modalities discussed since **T1**. The second rule introduces n to stand for number symbols, v as variable symbol, and symbols for arithmetic operations and exponentiation. The third rule generates equality, inequality, and value comparisons, while $x \backsim \mathit{pdf}$ serves to write that x follows a particular probability density function. The rest of the rules, from b onwards, look exactly the same as in **T4**, but obviously allow new expressions to be written.

Semantic Mapping. It is clear from the above how symbols are mapped to objects in the semantic domain.

Consequence Relation. The consequence relation in **T5** is $\mid\!\sim_{\mathfrak{4}}$ extended to detect inconsistency between constraints on numerical variables. This is handled by adding a macro that generates axioms that allow inconsistency to be deduced when constraints on numerical variables are such that there are no possible assignments of values to these variables which ensure that all constraints are satisfied.

Definition 9.1. For $\Pi \subseteq_\tau \mathcal{L}_{\mathfrak{5}}$ and $\phi \in \mathcal{L}_{\mathfrak{5}}$, the **consequence relation** $\mid\!\sim_{\mathfrak{5}}$ is such that:

- $\Pi \mid\!\sim_{\mathfrak{5}} \phi$ if $\phi \in \Pi$, or

- $\Pi \mid\!\sim_{\mathfrak{s}} x$ if $\forall 1 \leq n$, $\Pi \mid\!\sim_{\mathfrak{s}} \phi_i$ and $\mathbf{k}((\bigwedge_{i=1}^{n} \phi_i) \rightarrow x) \in \Pi$,
- $\mid\!\sim_{\mathfrak{s}}$ satisfies the *preference conflict* axioms:

$$\forall \Pi_i, \Pi_j \subseteq \Pi, \ \{\Pi_i \succ \Pi_j, \Pi_j \succ \Pi_i\} \rightarrow \bot, \tag{9.110}$$
$$\forall \Pi_i, \Pi_j \subseteq \Pi, \ \{\Pi_i \succeq \Pi_j, \Pi_j \succ \Pi_i\} \rightarrow \bot, \tag{9.111}$$

- $\mid\!\sim_{\mathfrak{s}}$ satisfies the *quantitative inconsistency axioms*, generated by the following macro:

> For all $\Pi_k \subseteq \Pi$ in which there are expressions over rational variables $v_1, \ldots, v_m$, for $m \geq 1$ in Π, if there is no assignment of values to all variables $v_1, \ldots, v_m$ which satisfies all constraints on these variables in Π_k, then there is an axiom $\mathbf{k}((\bigwedge \Pi_k) \rightarrow \bot)^{\mathsf{M}} \in \mathcal{L}_{\mathfrak{s}}$.

■

Remarks on soundness and completeness made for $\mid\!\sim_{\mathfrak{1}}$ and $\mid\!\sim_{\mathfrak{2}}$ apply for $\mid\!\sim_{\mathfrak{s}}$.

Keeping $\mid\!\sim_{\mathfrak{s}}$ simple and treating rational variables in an analogous way to natural language propositions are the main reasons for having $\mid\!\sim_{\mathfrak{s}}$ treat requirements over mathematical expressions in the same way as requirements over natural language propositions. Another approach would have been to check if constraints on a rational variable are not mutually exclusive, and if so, to block expressions over that variable from passing through $\mid\!\sim_{\mathfrak{s}}$. Doing this would also require that requirements over natural language propositions are blocked if they participate in conflicts, which was not the case in **T1**–**T4**. It would also mean that the consequence relation becomes much too specific, and less useful in the definitions of various derived relations.

9.3 Problem & Solution Concepts

The formulation of the requirements problem is not significantly affected by the introduction of rational variables and associated machinery. Quality constraints are now mentioned alongside goals which may be given at the very start of RE (as opposed to being elicited or otherwise obtained during RE).

Definition 9.2. Given a set of goals, quality constraints, domain assumptions, and agents, find the solution which ranks highest according to the chosen decision rule. ■

The decision rule remains a macro which produces a ranking of candidate solutions.

The main difference from solution concepts in **T4** comes from the new *quantitative operationalization function*. The original operationalization function (cf., Definition 5.4) from simpler Techne formalisms remains applicable in **T5**, but it is more appropriate to call it a *qualitative* operationalization function.

The quantitative operationalization relation returns all subsets of $\Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ from which ϕ can be deduced, where ϕ must be a requirement over rational

variables. It thereby gives all information in the requirements database which lead to rational variables taking values which satisfy some quality constraint.

Definition 9.3. Let $\mathcal{L}^z_{\mathbf{5}}$ be the set of expressions of $\mathcal{L}_{\mathbf{5}}$ such that every expression is either a quality constraint, or a domain assumption, or a task over rational variables. The **quantitative operationalization function**

$$\mathsf{Ov} : (\mathcal{L}^z_{\mathbf{5}} \cap \Delta) \longrightarrow \wp\left(\wp\left(\bigcup_{\forall \mathbf{x},\mathbf{y}} \Delta^{\mathbf{y}}_{\mathbf{x}}\right)\right) \tag{9.112}$$

for $\mathbf{x} \in \{\mathbf{k}, \mathbf{t}\}$ and $\mathbf{y} \in \{\textsc{m}, \textsc{p}, \textsc{h}\}$.

Let $\forall i, n_i$ be a rational number and v_i be a variable taking its value among rational numbers. Let there be m such variables v_i in ϕ, so that $1 \leq i \leq m$. Ov is defined as follows:

$\Pi \in \mathsf{Ov}(\phi \in \mathcal{L}^z_{\mathbf{5}})$ if and only if:

1. $\Pi \not\vdash_{\mathbf{5}} \bot$, i.e., Π is consistent;
2. $\{\mathbf{x}(v_i = n_i)^{\mathbf{y}} \in \mathcal{L}^z_{\mathbf{5}} \mid 1 \leq i \leq m\} \subseteq \Pi$, i.e., Π includes requirements which give equate variables in ϕ to values;
3. $\forall i, 1 \leq i \leq m$, one of more of the following hold:
   (a) $\mathbf{x}(v_i = n_i)^{\mathbf{y}} \in \bigcup_{\forall \mathbf{x},\mathbf{y}} \Delta^{\mathbf{y}}_{\mathbf{x}}$, i.e., $\mathbf{x}(v_i = n_i)$ is among domain assumptions and tasks in Δ,
   (b) $\exists \Phi \in \mathsf{Ov}(\mathbf{x}(v_i = n_i)^{\mathbf{y}})$ s.t. $\Phi \subseteq \Pi$, i.e., Π includes a quantitative operationalization of $\mathbf{x}(v_i = n_i)^{\mathbf{y}}$,
   (c) $\exists \Phi \in \mathsf{Op}(\mathbf{x}(v_i = n_i)^{\mathbf{y}})$ s.t. $\Phi \subseteq \Pi$, i.e., Π includes a qualitative operationalization of $\mathbf{x}(v_i = n_i)^{\mathbf{y}}$,
4. and if every variable v_i in ϕ is replaced with the constant n_i, then the mathematical expression in ϕ holds.

Every member Π of $\mathsf{Ov}(\phi)$ is a consistent set which includes all requirements which (i) are operationalized, (ii) assign values to every rational variable $v_1, \ldots, v_m$ in ϕ, and (iii) assign such constants $n_1, \ldots, n_m$ to the variables $v_1, \ldots, v_m$ that the functional relation specified over these variables in ϕ holds. $\mathsf{Ov}(\phi)$ thereby states all combinations of requirements which assign such constants to all variables in ϕ that ϕ is satisfied. ■

The qualitative operationalization function is used in the following definition of the threshold solution concept.

Definition 9.4. A **threshold solution** to the requirements problem given by a requirements database $\Delta \subseteq \mathcal{L}_{\mathbf{5}}$ is a set $S \subseteq_\tau \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ of domain assumptions and tasks, which satisfies the following four properties:

1. *Consistency:* $S \not\vdash_{\mathbf{s}} \bot$, i.e., all requirements in a threshold solution must be consistent;

2. *Qualitative threshold achievement:* $\forall \phi \in \Delta^{\mathsf{M}}_{\mathbf{g}},\ \exists \Pi \in \mathsf{Op}(\phi)$ s.t. $\Pi \subseteq S$, i.e., a threshold solution must include at least one operationalization of every Mandatory goal;

3. *Quantitative threshold achievement:* $\forall \phi \in \Delta^{\mathsf{M}}_{\mathbf{q}},\ \exists \Gamma \in \mathsf{Ov}(\phi)$ such that $\Gamma \subseteq S$, i.e., S must include a quantitative operationalization of every Mandatory quality constraint;

4. *Conformity:* $\forall \phi \in \Delta^{\mathsf{M}}_{\mathbf{k}} \cup \Delta^{\mathsf{M}}_{\mathbf{t}},\ S \mathrel{|\!\sim_4} \phi$, i.e., a threshold solution must satisfy all Mandatory domain assumptions and all Mandatory tasks;

5. *Minimality:* $\nexists S' \subset S$ s.t. S' satisfies the Consistency, Threshold achievement, and Conformity conditions, i.e., a threshold solution must include only the requirements necessary to satisfy the three conditions above.

■

The threshold solution concepts in **T4** and **T5** differ only in that the latter has the Quantitative threshold achievement condition, and renames the Threshold achievement condition into Qualitative threshold achievement. This reflects the introduction of quality constraints, and the possibility that the requirements database includes Mandatory quality constraints.

Just as in **T4**, an expanded threshold solution is a candidate solution.

Definition 9.5. A **candidate solution** to the requirements problem given by a requirements database $\Delta \subseteq \mathcal{L}_{\mathbf{5}}$ is a set $S \subseteq_\tau \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ of domain assumptions and tasks, which satisfies the following four properties:

1. *Threshold inclusion*: $\exists S' \subseteq_\tau S$, where S' is a threshold solution to the requirements problem in Δ, i.e., S is equivalent or superset of a threshold solution;

2. *Expansion:* there is a potentially empty $\Pi \subseteq \Delta_{\mathbf{k}} \cup \Delta_{\mathbf{t}}$ such that $S = S' \cup \Pi$, i.e., there is a potentially empty set of domain assumptions and tasks which are not Mandatory and which are included in S.

A candidate solution is either equal to a threshold solution (when Π is empty) or is a threshold solution which was expanded by adding non-Mandatory tasks and domain assumptions to it. ■

9.4 Derived Relations

T5 keeps all relations from **T4**, and adds the quantitative conflict relation.

9.4.1 Quantitative Conflict Relations

A set of expressions of **T5** can be such that it gives constraints on values of rational variables, yet there is no possible assignment of values to all these variables that satisfies these constraints. All expressions in that set are then said to be in the quantitative conflict relation.

Definition 9.6. There is a **quantitative conflict relation** between all members of $\Pi \subseteq \mathcal{L}_{\mathbf{5}}$ if and only if:

1. There are constraints on values of rational variables $v_1, \ldots, v_n$, for $n \geq 1$ in the expressions in Π,
2. There is no assignment of values to all variables $v_1, \ldots, v_n$ which satisfies the constraints in the expressions in Π,
3. There is no $\Pi' \subset \Pi$ for which the conditions 1 and 2 above hold.

■

Quantitative conflict can be specialized onto Type-A, Type-B, and Type-C quantitative conflict in the straightforward way, by analogy to the specialization of the conflict relation in **T1**.

9.5 Database Interface

The database interface in **T5** keeps all filters from **T4** and adds filters specific to quality constraints. These additional filters follow those defined for goals, as quality constraints can be seen as goals over rational variables. The new filters are as follows:

- All top-level quality constraints:

$$\mathsf{AQTop}(\Delta) = \{\phi \mid \phi \in \Delta_{\mathbf{q}} \text{ and } \nexists \psi \in \Delta,\ \phi \in \bigcup \mathsf{Op}(\psi)\} \tag{9.113}$$

- All quality constraints that have at least n qualitative operationalizations:

$$\mathsf{AQnOp}(\Pi) = \{\phi \mid \phi \in \Pi_{\mathbf{q}} \text{ and } |\mathsf{Op}(\phi)| \geq \mathsf{n}\} \tag{9.114}$$

- All quality constraints that have at least n quantitative operationalizations:

$$\mathsf{AQnOv}(\Pi) = \{\phi \mid \phi \in \Pi_{\mathbf{q}} \text{ and } |\mathsf{Ov}(\phi)| \geq \mathsf{n}\} \tag{9.115}$$

9.6 Discussion

This section focuses on how **T5** can be used to model information that was recognized as crucial in the research into the relaxation of requirements [20, 34, 4] and the evaluation of their partial satisfaction [20].

Fuzzy Relaxation. Baresi et al. associate every fuzzy operator with a predefined membership function. E.g., if there is a quality constraint $\mathbf{q}(v < 6\text{hrs})$ here (a goal $G(v < 6\text{hrs})$ for them, as they allow quantitative variables in goals), then relaxing it would amount to replace it with $\mathbf{q}(v <_f 6\text{hrs})$ (in their notation, $\mathcal{G}(v <_f 6\text{hrs})$), where $\leq_f$ is a fuzzy operator. Their interpretation of $v <_f 6\text{hrs}$ is that there is a fuzzy membership function μ which returns the level of satisfaction as a function of v, and the shape of μ is predefined (for $<_f$, it is positive and

constant until $v = 6$, then decreases up to the satisfaction value 0 for some $v > 6$). The operator $<_f$ can be defined in **T5** by reproducing in a domain assumption, a function which has the form of the fuzzy membership function for $<_f$, as defined by Baresi et al. One can define as follows a macro which takes a fuzzy goal of the form $\mathcal{G}(v <_f n)$, with $n \in \mathbb{R}$, and transforms it into requirements that can be added to a requirements database Δ in **T5**:

$\forall \mathcal{G}(v <_f n)$ `where` $v \in V$ `and` $n \in \mathbb{R}$
`add` $\{\mathtt{k}(v' = \mu(v))\}$ `to` Δ, `and apply the softgoal macro on` Δ `and` v,

where v' is a rational variable, i.e., $v \in V$, and μ is a function which is defined according to the function pattern defined by Baresi et al. for the fuzzy operator $<_f$. It is straightforward to define similar macros for all other fuzzy operators defined by Baresi et al.

Baresi et al. also define binary connectives, such as fuzzy conjunction. These can be defined here as well, as functions of variables defined using fuzzy membership functions. Each binary fuzzy operator gives one function specified in a domain assumption and using an approximation relation. In the formalism from Baresi et al., one way to define fuzzy conjunction $\wedge_f$ between two variables v_1 and v_2, each of which has an accompanying fuzzy membership function $\mu(v_1)$ and $\mu(v_2)$, is as follows: $\mu(v_1 \wedge_f v_2) = \mu(v_1) \cdot \mu(v_2)$. In **T5**, $\mu(v_1)$ and $\mu(v_2)$ give satisfaction levels. The fuzzy conjunction connective between two quality constraints, respectively over variables v_1 and v_2 is introduced in Δ as the domain assumption $\mathtt{k}(v_3 = \mu(v_1) \cdot \mu(v_2))$, where v_3 is the joint level of satisfaction over variables v_1 and v_2.

Probabilistic Relaxation. To handle idealistic requirements, Letier & van Lamsweerde [20] suggest the association of probability estimates to constraints on quantitative variables. This is allowed in **T5**, and has been illustrated earlier (cf., §9.1).

10 Case Study

The aim in this section is to discuss the features of the Techne formalisms in the context of a complete case study obtained from an European information technology consultancy.

10.1 Background

A large Asian electronics company designs, manufactures, and distributes mobile phones and tablet devices in all European countries.

The company asked the consultancy to engineer a system which should manage orders from shops, data collection by merchandisers, the synthesis and reporting of merchandising data, and information distribution to and sharing between salespeople.

These four rough requiremenets fit together as follows. When the company delivers some quantity of a new product to a country, shops order these products. The company itself has no shops. The system should handle these orders, mainly by allowing shop managers to make and track orders. After a product is delivered to, anywhere from hundreds to thousands of shops, the company needs to assess how well the product is marketed in these shops, as this directly influences sales numbers. To do so, the company sends at a certain frequency its merchandisers to visit every shop. A merchandiser visits a shop, observes how the company's products are marketed, talks to salespeople at the shop, and fills out a questionnaire. The system should deliver questions and collect answers remotely, as the company's plan is for every merchandiser to carry a tablet device enabled for mobile Internet access. The system should allow company's employees to view the data collected at the shops, and produce reports showing different subsets of that data. Based on the analysis of the data, the company wishes to use the system to inform and train salespeople. The system should allow the company to send out information about products and otehrwise to salespeople working in shops.

Requirements were given verbally at meetings with several managers and engineers of one European branch of the company. There were no legacy systems to which the new system should connect, or which otherwise influence the engineering of the future system. Requirements given initially were severely incomplete and imprecise. The company expected of the consultancy to do requirements engineering in iterations and to assist the company at each iteration in brainstorming about the goals, functionalities, and other characteristics and behaviors of the system-to-be. In other words, requirements engineering had to help explore the problem and solution space. It had to use lightweight notations, as early solutions or parts thereof could end up rejected at later iterations. Finally, it had to use a notation that can be understandable to the client's personnel, without having to train them in the specifics of a notation.

In the rest of this section, the system-to-be is called the Shop Network Management (SNM) system.

10.2 Building up the Requirements Database

High-level goals of SNM are:

| $\mathbf{g}(p_1$: Data can be collected at shops)$^{\mathsf{M}}$
| $\mathbf{g}(p_2$: Reports can be created from data)$^{\mathsf{M}}$
| $\mathbf{g}(p_3$: Tasks can be allocated to merchandisers)$^{\mathsf{M}}$
| $\mathbf{g}(p_4$: Merchandisers can be supervised)$^{\mathsf{M}}$
| $\mathbf{g}(p_5$: Salesmen can receive information)$^{\mathsf{M}}$
| $\mathbf{g}(p_6$: Shops can order products)$^{\mathsf{M}}$
| $\mathbf{g}(p_7$: Shops can track product orders)$^{\mathsf{M}}$
| $\mathbf{g}(p_8$: Salesmen can access question and answer database)$^{\mathsf{M}}$

The company required that reports need to be customizable. More specifically, this means that instead of predefining a set of report templates and hardcoding

them, templates should be definable through the system. It was considered as crucial that the SNM is responsive, easy to use, and that it can appropriately handle exceptions. An example of an exception is the situation when the merchandiser is at a shop where she has no fast Internet access, yet she wishes to add images to the report about that shop. Since all these additional expectations involve vagueness, the following softgoals are added to the database:

| $\mathbf{sg}(\tilde{q}_1$: Customizable reports)
| $\mathbf{sg}(\tilde{q}_2$: User interfaces quickly respond to input)
| $\mathbf{sg}(\tilde{q}_3$: Easy to use)
| $\mathbf{sg}(\tilde{q}_4$: Appropriately handle exceptions)

Broad constraints are imposed on how the goals can be satisfied. As they amount to assumptions about what is (not) feasible, the constraints are captured as domain assumptions.

| $\mathbf{k}(p_9$: Every merchandiser is given a tablet device which is enabled for mobile Internet access)$^{\mathsf{M}}$
| $\mathbf{k}(p_{10}$: There always exist shops in regions without third generation mobile Internet access and without Wi-Fi Internet access)$^{\mathsf{M}}$
| $\mathbf{k}(p_{11}$: There can exist products which are sold in shops, but are not in the product database)$^{\mathsf{M}}$

The softgoals will give preferences over alternative refinements and operationalizations of the goals. The softgoals thus remain only recorded for the moment, without being used. The next step is to add details to goals. One way to do so would be to allocate the goals to roles. The following roles are relevant:

| R_1: Merchandiser
| R_2: Supervisor of merchandisers
| R_3: Analyst at the company
| R_4: SNM web software
| R_5: SNM mobile software
| R_6: Shop manager
| R_7: Shop salesman

The problem is that the goals are not detailed enough to make their allocation to roles. E.g., it may seem reasonable to add $\mathsf{R}_1\mathbf{g}(p_1)$, but this is not adequate: collection of data at shops involves four roles, not one – the Marchandiser who will be collecting data, the SNM web software which delivers guidelines for data collection (namely, the questionnaire), the SNM mobile software which the Merchandiser uses to input data, and the Supervisor of merchandisers, who needs to define the questionnaires and allocate questionnaires and shops to Merchandisers. It is more useful to focus on adding details to the goals first, then allocate them to roles.

Collection of data at shops ($\mathbf{g}(p_1)^{\mathsf{M}}$). Discussing $\mathbf{g}(p_1)$ leads to new goals:

| $\mathbf{g}(p_{12}$: Merchandiser can access the SNM mobile software)$^{\mathsf{H}}$
| $\mathbf{g}(p_{13}$: SNM mobile software and SNM web software can exchange data)$^{\mathsf{H}}$

$\mathbf{g}(p_{14}$: Personal task list can be viewed$)^{\mathsf{H}}$
$\mathbf{g}(p_{15}$: Each task in a personal task list instructs a visit to a shop$)^{\mathsf{M}}$
$\mathbf{g}(p_{16}$: Shop questionnaire can be accessed$)^{\mathsf{M}}$
$\mathbf{g}(p_{17}$: Shop questionnaire can be filled out$)^{\mathsf{M}}$
$\mathbf{g}(p_{18}$: Filled-out questionnaire can be uploaded to the SNM web software$)^{\mathsf{M}}$

The goals above refine $\mathbf{g}(p_1)$, so that the following axiom is added:

$$\mathbf{k}\left(\bigwedge_{i=12}^{14}\mathbf{g}(p_i)^{\mathsf{H}} \wedge \bigwedge_{j=15}^{18}\mathbf{g}(p_j)^{\mathsf{M}} \rightarrow \mathbf{g}(p_1)^{\mathsf{M}}\right)^{\mathsf{M}}$$

The goals in the refinement are either mandatory like $\mathbf{g}(p_1)^{\mathsf{M}}$, or inherit the mandatory modality from that goal. The reason for having different optionality modalities on the goals in the refinement, is that changes to the optionality modality of $\mathbf{g}(p_1)$ should not affect some of the goals in a refinement. The goals $\mathbf{g}(p_{16})$, $\mathbf{g}(p_{17})$ and $\mathbf{g}(p_{18})$ are crucial, independently of the refined goal. It is important to see that while $\mathbf{g}(p_1)$ is refined by these goals, and they have been identified by investigating $\mathbf{g}(p_1)$, goals in a refinement (and more generally, requirements in a premise of an argument) can be of interest independently from the goal that they refine (i.e., the conclusion of an argument). While the Inherited modality can be used on all members of a refinement, it need not always be used over all members of all refinements.

Softgoals and preferred goals on data collection at shops. New goals suggest new softgoals, to say more about how well the goals ought to be satisfied; in addition, a preferred goal was suggested:

$\mathbf{sg}(\tilde{q}_5$: Communication is secure between SNM web and mobile software)
$\mathbf{sg}(\tilde{q}_6$: Minimize fictional shop reports sent without having visited the shop)
$\mathbf{g}(p_{19}$: Supervisor of merchandisers and Merchandiser can communicate about tasks using the SNM web and mobile software$)^{\mathsf{P}}$

Problem of fictional shop reports ($\mathbf{sg}(\tilde{q}_6)$). The softgoal $\mathbf{sg}(\tilde{q}_6)$ reflects the case of a merchandiser filling out a report about a shop without having visited that shop, or basing the report on a past visit to that shop (a visit for which a report was already filed). The following domain assumptions can be used to state the problem in the requirements database:

$\mathbf{k}(p_{20}$: There is no verification of the Merchandiser having visited a shop before sending a report for that shop$)^{\mathsf{P}}$
$\mathbf{k}(p_{21}$: Fictional reports can be sent to SNM$)^{\mathsf{P}}$

$$\mathbf{k}\left(\mathbf{k}(p_{20})^{\mathsf{P}} \rightarrow \mathbf{k}(p_{21})^{\mathsf{P}}\right)^{\mathsf{M}}$$

Making $\mathbf{k}(\mathbf{k}(p_{20})^{\mathsf{P}} \rightarrow \mathbf{k}(p_{21})^{\mathsf{P}})$ mandatory indicates that it is inevitable that if there is no verification of shop visits, then there will be fictional reports. Making the antecedent and the consequent preferred indicates that while they can be satisfied, there can also be solutions which violate them.

First option to avoid fictional reports. There are at least two ways to solve the problem of fictional reports, and each is captured by inference relations. One option is to require Shop managers to confirm a shop visit through the SNM web software:

$\mathbf{g}(p_{22}$: Shop manager confirms shop visit by Merchandiser$)^{\mathsf{H}}$

$\mathbf{g}(p_{23}$: Shop manager can use SNM web software to confirm shop visit by Merchandiser$)^{\mathsf{H}}$

$$\mathbf{k}\left(\mathbf{g}(p_{22})^{\mathsf{H}} \wedge \mathbf{g}(p_{23})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{17})^{\mathsf{M}}\right)^{\mathsf{M}}$$

Since this is intended to ensure that $\mathbf{k}(p_{21})^{\mathsf{P}}$ is avoided, there is also the conflict:

$$\mathbf{k}\left(\mathbf{g}(p_{22})^{\mathsf{H}} \wedge \mathbf{k}(p_{20})^{\mathsf{P}} \rightarrow \bot\right)^{\mathsf{M}}$$

Second option to avoid fictional reports. Another option is to determine by GPS the location of the Merchandiser, when she requests to access a questionnaire, and allow the Merchandiser to fill out the questionnaire if she is within some distance of the shop location.

$\mathbf{k}(p_{24}$: Location can be determined by the tablet device using GPS$)^{\mathsf{H}}$

$\mathbf{t}(p_{25}$: Merchandiser requests access to shop questionnaire$)^{\mathsf{H}}$

$\mathbf{g}(p_{26}$: Current location known$)^{\mathsf{H}}$

$$\mathbf{k}\left(\mathbf{k}(p_{24})^{\mathsf{H}} \wedge \mathbf{t}(p_{25})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{26})^{\mathsf{H}}\right)^{\mathsf{M}}$$

$\mathbf{q}($ LocErr $\leq$ 10m, where LocErr is the number of meters between the current location reported by GPS of tablet device and the location recorded for a shop$)^{\mathsf{H}}$

$\mathbf{t}(p_{27}$: Compute LocErr$)^{\mathsf{H}}$

$$\mathbf{k}\left(\mathbf{g}(p_{26})^{\mathsf{H}} \wedge \mathbf{t}(p_{27})^{\mathsf{H}} \wedge \mathbf{q}(\text{LocErr} \leq 10\text{m})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{17})^{\mathsf{M}}\right)^{\mathsf{M}}$$

There is again a conflict, to indicate that $\mathbf{k}(p_{21})^{\mathsf{P}}$ is avoided if current location is known, the value of LocErr is computed and is is lower than the limit set in the quality constraint:

$$\mathbf{k}\left(\mathbf{g}(p_{26})^{\mathsf{H}} \wedge \mathbf{t}(p_{27})^{\mathsf{H}} \wedge \mathbf{q}(\text{LocErr} \leq 10\text{m})^{\mathsf{H}} \wedge \mathbf{k}(p_{20})^{\mathsf{P}} \rightarrow \bot\right)^{\mathsf{M}}$$

Relaxation of LocErr limit. The 10m limit is arbitrary in the quality contraint $\mathbf{q}(\text{LocErr} \leq 10\text{m})^{\mathsf{H}}$. Some shops can be big enough that the quality constraint is violated even if the Merchandiser is in fact in the shop. This problem can be made explicit in the database as follows:

$\mathbf{k}(p_{28}$: Surface plans are not known for each shop.$)^{\mathsf{M}}$

$\mathbf{k}(p_{29}$: There is no unique appropriate limit value below 1000m for LocErr.$)^{\mathsf{M}}$

$$\mathbf{k}\left(\mathbf{k}(p_{28})^{\mathsf{M}} \rightarrow \mathbf{k}(p_{29})^{\mathsf{M}}\right)^{\mathsf{M}}$$

Macro 1 begins.
For each $\mathbf{q}(\text{LocErr } X\ Y\text{m})^Z$ such that $X \in \{<, =, >, \leq, \neq, \geq\}$, $Y \in \mathbb{Q}^{\geq 0}$, and $Z \in \{\textsf{M}, \textsf{P}, \textsf{H}\}$, add

$$\mathbf{k}\left(\mathbf{k}(p_{29})^{\textsf{M}} \wedge \mathbf{q}(\text{LocErr } X\ Y\text{m})^Z \rightarrow \bot\right)^{\textsf{M}}$$

to Δ.
Macro 1 ends.

The macro indicates that there will be a conflict between $\mathbf{k}(p_{29})^{\textsf{M}}$ and any quality constraint on LocErr. $\mathbb{Q}^{\geq 0}$ is the set of rational numbers greater than, and including 0.

Because $\mathbf{k}(p_{29})^{\textsf{M}}$ is mandatory, an option to avoid the conflicts generated by the macro above is to avoid having quality constraints on the value of LocErr. To do so, relax all quality constraints on LocErr. Between fuzzy and probabilistic relaxation, fuzzy relaxation seems more appropriate in this case. Probabilistic relaxation would constrain the frequency of the Merchandiser being beyond some LocErr value (i.e., being "too far" from a shop). Fuzzy relaxation instead gives an entirely subjective function which returns a satisfaction level for a particular LocErr value. Although the shape of the satisfaction function will remain based on subjective considerations, it can be agreed upon through the discussion with the system stakeholders. Suppose that the following shape was agreed on: the closer LocErr is to 0, the higher the satisfaction level (with maximum 1), but if LocErr is equal or greater than 500m, the satisfaction level is 0. The satisfaction function SatLocErr is defined as follows in the requirements database:

$$\mathbf{k}\left(\text{SatLocErr}(\text{LocErr}) = \begin{cases} 1 - \text{LocErr}/500 & \text{if LocErr} < 500; \\ 0 & \text{otherwise.} \end{cases}\right)^{\textsf{M}}$$

To complete fuzzy relaxation, a softgoal is needed over values of LocErr. That softgoal should be a macro, in order to serve for the comparison of alternative solutions. Now, remark that the system-to-be will have to record LocErr for every Merchandiser and for every visit to a shop. If different solutions to the requirements problem will affect LocErr differently, then these differences should be identifiable from comparing means and variances of over some sample of LocErr values. Lower means and lower variances are obviously preferred, leading to the following softgoals and the corresponding macros which generate preference relations. It is assumed for simplicity that LocErr values are equally probable; also, S_j and S_k are arbitrary candidate solutions to the requirements problem.

$\mathbf{sg}(\tilde{q}_7$: Low $E[\text{LocErr}] = \frac{1}{n_w}\sum_{i=1}^{n_w} \text{LocErr}_i$, where n_w is the number of recorded LocErr values during w-th week)

$\mathbf{sg}(\tilde{q}_8$: Low $\text{Var}(\text{LocErr}) = \frac{1}{n_w}\sum_{i=1}^{n_w} \text{LocErr}_i^2 - E[\text{LocErr}]^2$, where n_w is the number of recorded LocErr values during w-th week)

Macro 2 begins.
$\mathbf{sg}(\tilde{q}_7) \stackrel{def}{=}$ (
Let $x_j \stackrel{def}{=} \textsc{val}(S_j, E[\text{LocErr}])$ and $x_k \stackrel{def}{=} \textsc{val}(S_k, E[\text{LocErr}])$,

- if $\text{SatLocErr}(x_j) \geq \text{SatLocErr}(x_k)$, then add $\{\mathbf{k}(E[\text{LocErr}] = x_k)\} \succeq \{\mathbf{k}(E[\text{LocErr}] = x_j)\}$ to Δ,
- if $\text{SatLocErr}(x_j) > \text{SatLocErr}(x_k)$, then add $\{\mathbf{k}(E[\text{LocErr}] = x_k)\} \succ \{\mathbf{k}(E[\text{LocErr}] = x_j)\}$ to Δ,
- if $\text{SatLocErr}(x_j) = \text{SatLocErr}(x_k)$, then add $\{\mathbf{k}(E[\text{LocErr}] = x_k)\} \approx \{\mathbf{k}(E[\text{LocErr}] = x_j)\}$ to Δ.)

Macro 2 ends.

Macro 3 begins.
$\mathbf{sg}(\tilde{q}_8) \stackrel{def}{=}$ (
Let $x_j \stackrel{def}{=} \textsc{val}(S_j, \text{Var}[\text{LocErr}])$ and $x_k \stackrel{def}{=} \textsc{val}(S_k, \text{Var}[\text{LocErr}])$,

- if $\text{SatLocErr}(x_j) \geq \text{SatLocErr}(x_k)$, then add $\{\mathbf{k}(\text{Var}[\text{LocErr}] = x_k)\} \succeq \{\mathbf{k}(\text{Var}[\text{LocErr}] = x_j)\}$ to Δ,
- if $\text{SatLocErr}(x_j) > \text{SatLocErr}(x_k)$, then add $\{\mathbf{k}(\text{Var}[\text{LocErr}] = x_k)\} \succ \{\mathbf{k}(\text{Var}[\text{LocErr}] = x_j)\}$ to Δ,
- if $\text{SatLocErr}(x_j) = \text{SatLocErr}(x_k)$, then add $\{\mathbf{k}(\text{Var}[\text{LocErr}] = x_k)\} \approx \{\mathbf{k}(\text{Var}[\text{LocErr}] = x_j)\}$ to Δ.)

Macro 3 ends.

Values for $E[\text{LocErr}]$ and $\text{Var}[\text{LocErr}]$ are straightforward to compute, when given a sample of LocErr values. The mechanism which will generate these values in the system-to-be are the actual visits made to shop. During RE, this process can be simulated. This in turn requires domain assumptions which satisfy two conditions. Firstly, no two of them should be in the same candidate solution, for doing otherwise would require specifying when each assumption applies (which can be done in Techne by making each of these domain assumptions conclusions of an argument and having applicability conditions in the premises of the argument, but adds unnecessary complications in the present discussion). This first condition is achieved by adding conflict relations between domain assumptions that generate LocErr values. Secondly, each of these domain assumptions needs to return LocErr values. This can be accomplished in many different ways, including:

- Making LocErr follow a probability distribution function (i.e., LocErr is treated as a random variable), e.g., a normal distribution with mean of $50m$ and variance of $20m^2$: $\mathbf{k}(\text{LocErr} \backsim \mathcal{N}(50m, 20m^2))$. this means that if two candidate solutions should significantly differ over LocErr mean and variance, then it is needed to have LocErr follow different probability distribution functions in each candidate solution.
- Making LocErr a function of other variables, whereby these other variables

may include random variables. This allows for a potentially elaborate model to be defined, which would give LocErr values. E.g., ErrLoc = $f(v_1, \ldots, v_n)$, where each v_j in some subset of these variables would be a binary variable indicating the kind of environment in which the shop is located. As shop questionnaires include a multiple choice question which asks for a description of the shop environment, a predictive model could be fit to past data, where the estimated coefficient of each of these variables would indicate the influence of location on LocErr. This would in turn reflect the company analysts' intuition that the "nicer" the environment of a shop, the less likely the shop will be surveyed and the more likely the questionnaire would be filled out outside the shop (and in that "nicer" environment). Candidate solutions may then differ in how they influence the values of the coefficients in the predictive model, and thereby result in different ErrLoc values.

Regardless of the mechanism chosen to simulate the values of ErrLoc, adding it results in a quantitative operationalization relation between the domain assumption stating that mechanism and the domain assumption which assigns a value to ErrLoc.

Suppose that it is decided to treat LocErr as a random variable, and that it should follow a normal distribution in every candidate solution. At this point, no complete candidate solution is known. It can, however, be assumed that if a policy is adopted to randomly verify if a Merchandiser visited a shop, then the mean of LocErr should be lower than if that policy is not adopted. Every candidate solution which satisfies that policy should then have lower mean and variance than a candidate which does not satisfy the same policy. The result of this discussion is to add the following to the requirements database:

$\mathbf{g}(p_{30}$: Perform random verifications of shop visits$)^{\mathsf{P}}$

$\mathbf{k}($ LocErr $\backsim \mathcal{N}(35m, 15m^2))^{\mathsf{P}}$

$\mathbf{k}($ LocErr $\backsim \mathcal{N}(50m, 20m^2))^{\mathsf{P}}$

$$\mathbf{k}\left(\mathbf{k}(p_{30})^{\mathsf{P}} \rightarrow \mathbf{k}(\text{LocErr} \backsim \mathcal{N}(35m, 15m^2))^{\mathsf{P}}\right)^{\mathsf{M}}$$

$$\mathbf{k}\left(\mathbf{k}(\text{LocErr} \backsim \mathcal{N}(35m, 15m^2))^{\mathsf{P}} \wedge \mathbf{k}(\text{LocErr} \backsim \mathcal{N}(50m, 20m^2))^{\mathsf{P}} \rightarrow \bot\right)^{\mathsf{M}}$$

The requirements above give two alternative distributions for LocErr, and say that the mean and variance will be lower if $\mathbf{g}(p_{30})^{\mathsf{P}}$ is satisfied. When sample values of LocErr are generated by following these probability distribution functions, the values of E[LocErr] and Var[LocErr] can be computed according to each distribution function, and the Macros 2 and 3 applied to obtain preference preference relations. As the rule of large numbers applies, it is not necessary to generate values. Rather, the mean and variance can be compared, resulting in the following preference relations:

$\{\mathbf{k}(E[\text{LocErr}] = 35m)^{\mathsf{P}}\} \succ \{\mathbf{k}(E[\text{LocErr}] = 50m)^{\mathsf{P}}\}$

$\mathbf{sg}(\tilde{q}_7) \leftarrow \{$
$\quad \{\mathbf{k}(E[\text{LocErr}] = 35m)^{\mathsf{P}}\} \succ \{\mathbf{k}(E[\text{LocErr}] = 50m)^{\mathsf{P}}\}$
$\quad \}$
$\{\mathbf{k}(\text{Var}[\text{LocErr}] = 15m^2)^{\mathsf{P}}\} \succ \{\mathbf{k}(\text{Var}[\text{LocErr}] = 20m^2)^{\mathsf{P}}\}$
$\mathbf{sg}(\tilde{q}_8) \leftarrow \{$
$\quad \{\mathbf{k}(\text{Var}[\text{LocErr}] = 15m^2)^{\mathsf{P}}\} \succ \{\mathbf{k}(\text{Var}[\text{LocErr}] = 20m^2)^{\mathsf{P}}\}$
$\quad \}$

Conditions to fill out a shop questionnaire. The discussions above which focused on the fictional shop reports, namely on how to minimize their number. It was required that there be a visit to a shop before it becomes possible for the Merchandiser to fill out a questionnaire. The first option required that verification be manual, as it asks that the Shop manager can satisfy $\mathbf{g}(p_{22})^{\mathsf{H}}$ and that the Shop manager can do so via the SNM web software. Both these goals are delimited enough that they can be allocated to the said roles:
$(\mathsf{R}_6\mathbf{g}(p_{22})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_4\mathbf{g}(p_{23})^{\mathsf{H}})^{\mathsf{H}}$

The second option is to use GPS to determine the location of the Merchandiser when she requests to access the shop questionnaire. The distance to shop location, i.e., the value of LocErr, plays a key role in this approach, as it is not feasible to ask the Merchandiser to be at the exact coordinates known for a shop, but within some distance of the shop. The task $\mathbf{t}(p_{27})^{\mathsf{H}}$ of computing the value of LocErr is the responsibility of SNM mobile software:
$(\mathsf{R}_5\mathbf{t}(p_{27})^{\mathsf{H}})^{\mathsf{H}}$

Revisiting the Inherited modality. Observe that the responsibility allocations above are over requirements which carry the Inherited optionality modality, and that as a result, responsibility allocations themselves must carry that same modality. It is, however, not clear from the relations introduced up to this point which optionality modality these responsibility allocations would obtain. The rule which is not part of Techne, but which can be applied is that an agency, responsibility, or commitment relation, which has the Inherited optionality modality inherits the optionality modality of the requirement which it ties to the role or agent. If applied, this rule makes, e.g., $\mathsf{R}_5\mathbf{t}(p_{27})^{\mathsf{H}}$ inherit the Mandatory modality if the chosen operationalization of $\mathbf{g}(p_{17})^{\mathsf{M}}$ includes $\mathbf{t}(p_{27})^{\mathsf{H}}$ (which itself will then inherit the Mandatory modality, by being in the premises of the argument for $\mathbf{g}(p_{17})^{\mathsf{M}}$).

Refining $\mathbf{g}(p_{12})^{\mathsf{M}}$. Access to SNM mobile software requires that the Merchandiser logs in, that the SNM web software has been configured by the Supervisor of merchandisers to allow the Merchandiser access, and that the SNM mobile software is running. This gives a straightforward refinement of $\mathbf{g}(p_{12})^{\mathsf{M}}$:

$\mathbf{g}(p_{31}$: SNM mobile software is running.$)^{\mathsf{H}}$
$\mathbf{g}(p_{32}$: Provide correct username and password at login.$)^{\mathsf{H}}$
$\mathbf{g}(p_{33}$: Merchandiser is allowed access to SNM mobile software.$)^{\mathsf{H}}$

$$\mathbf{k}\left(\mathbf{g}(p_{31})^{\mathsf{H}} \wedge \mathbf{g}(p_{32})^{\mathsf{H}} \wedge \mathbf{g}(p_{33})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{12})^{\mathsf{M}}\right)^{\mathsf{M}}$$

$(\mathsf{R}_5\mathbf{g}(p_{31})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_1\mathbf{g}(p_{32})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_2\mathbf{g}(p_{33})^{\mathsf{H}})^{\mathsf{H}}$

If the goal $\mathbf{g}(p_{12})^{\mathsf{M}}$ is made the responsibility of the Merchandiser, then dependency relations can be defined based on the refinement above. Since no actual agents who would occupy these roles are known at this time, a macro is used to add dependency relations after agents have been added to the requirements database.

$(\mathsf{R}_1\mathbf{g}(p_{12})^{\mathsf{M}})^{\mathsf{H}}$

`Macro 4 begins.`
For each A_i such that $\mathsf{O}(\mathsf{R}_1, \mathsf{A}_i) \in \Delta$, add

$$\mathbf{k}\left(\mathbf{g}(p_{31})^{\mathsf{H}} \wedge \mathbf{g}(p_{32})^{\mathsf{H}} \wedge \mathbf{g}(p_{33})^{\mathsf{H}} \rightarrow \mathsf{A}_i\mathbf{g}(p_{12})^{\mathsf{M}}\right)^{\mathsf{M}}$$

to Δ
`Macro 4 ends.`

An obvious exception which can occur and block the satisfaction of $\mathbf{g}(p_{12})^{\mathsf{M}}$ is if the Merchandiser forgets the credentials needed to login. This can be stated with a domain assumption and a Type-B conflict relation (cf., Definition 5.14), as follows:

$\mathbf{k}(p_{34}$: Merchandisers may forget login credentials.$)^{\mathsf{M}}$
$(\mathsf{R}_1\mathbf{k}(p_{34})^{\mathsf{M}})^{\mathsf{M}}$

$$\mathbf{k}\left(\mathbf{k}(p_{34})^{\mathsf{M}} \wedge \mathbf{g}(p_{32})^{\mathsf{H}} \rightarrow \bot\right)^{\mathsf{M}}$$

A straightforward two-step approach to resolve this conflict is firstly to add new requirements that there must be a procedure by which the Merchandiser can recover login credentials. The second step is then to remove both the domain assumption which acts as the blocker and the responsibility assignment of that domain assumption to the Merchandiser role. The two steps are:

1. Add the following goal and conflict to the requirements database:
   $\mathbf{g}(p_{35}$: Merchandiser can recover login credentials$)^{\mathsf{M}}$

   $$\mathbf{k}\left(\mathbf{g}(p_{35})^{\mathsf{M}} \wedge \mathbf{k}(p_{34})^{\mathsf{M}} \rightarrow \bot\right)^{\mathsf{M}}$$

2. Because the goal and conflict just added reflect the modeler's intention to eliminate the conflict due to the blocker $\mathbf{k}(p_{34})^{\mathsf{M}}$, both the blocker and the conflict relation are to be deleted from the requirements database – i.e., eliminate $\mathbf{k}(p_{34})^{\mathsf{M}}$, $\mathsf{R}_1(\mathbf{k}(p_{34})^{\mathsf{M}})^{\mathsf{M}}$, and $\mathbf{k}(\mathbf{k}(p_{34})^{\mathsf{M}} \wedge \mathbf{g}(p_{32})^{\mathsf{H}} \rightarrow \bot)^{\mathsf{M}}$.

Since there are best practices on how the recovery of login credentials should work, $\mathbf{g}(p_{35})^{\mathsf{M}}$ can be operationalized without adding too much detail early on in the RE process. Instead, a task is assigned to the Software engineer role, as follows:

$\mathbf{t}(p_{36}$: Implement best practice login credentials recovery process.$)^{\mathsf{H}}$

$$\mathbf{k}\left(\mathbf{t}(p_{36})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{35})^{\mathsf{M}}\right)^{\mathsf{M}}$$

R_8: Software engineer

$(\mathsf{R}_8\mathbf{t}(p_{36})^{\mathsf{H}})^{\mathsf{H}}$

Operationalization of $\mathbf{g}(p_{13})^{\mathsf{H}}$. Making it possible for SNM mobile and web software to exchange data requires the use of some infrastructure that would allow such data to be exchanged. There are two straightforward options, captured via two alternative operationalizations of $\mathbf{g}(p_{13})^{\mathsf{H}}$.

- Option 1 below consists of having the SNM mobile and web software exchange data over the Internet:

  $\mathbf{t}(p_{37}$: Implement data exchange using the Internet Protocol Suite.$)^{\mathsf{H}}$

  $\mathbf{t}(p_{38}$: Implement the SNM mobile software as a Google Android application.$)^{\mathsf{H}}$

  $\mathbf{t}(p_{39}$: Implement the SNM web software as a web application.$)^{\mathsf{H}}$

  $$\mathbf{k}\left(\mathbf{t}(p_{37})^{\mathsf{H}} \wedge \mathbf{t}(p_{38})^{\mathsf{H}} \wedge \mathbf{t}(p_{39})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{13})^{\mathsf{H}}\right)^{\mathsf{M}}$$

- Option 2 below is to exchange data by having the SNM mobile application connect – independently of the Internet – to the company's intranet:

  $\mathbf{t}(p_{40}$: Implement data exchange using a proprietary protocol.$)^{\mathsf{H}}$

  $\mathbf{t}(p_{41}$: Implement the SNM web software as an intranet server application which is not connected to the Internet.$)^{\mathsf{H}}$

  $$\mathbf{k}\left(\mathbf{t}(p_{38})^{\mathsf{H}} \wedge \mathbf{t}(p_{40})^{\mathsf{H}} \wedge \mathbf{t}(p_{41})^{\mathsf{H}} \rightarrow \mathbf{g}(p_{13})^{\mathsf{H}}\right)^{\mathsf{M}}$$

A contract relation is added to make sure these two options are alternatives, and that under no conditions would one be willing to implement them both:

$$\mathbf{k}\left(\mathbf{t}(p_{37})^{\mathsf{H}} \wedge \mathbf{t}(p_{40})^{\mathsf{H}} \rightarrow \bot\right)^{\mathsf{M}}$$

Observe that the conflict relation is between two alternatives (in the sense of Definition 5.12), making this a Type-A conflict (cf., Definition 5.13). It follows that we should not resolve it immediately (by choosing one of the alternatives and deleting the other) – the reason for avoiding resolution this early is that a qualitative tradeoff relation could be identified between these alternatives, and the choice of one over another thus becomes less straightforward. Recall that according to $\mathbf{sg}(\tilde{q}_5)$, communication between the SNM web and mobile software should be secure. This suggests strict preference for communication to happen via the company intranet, not through the Internet, that is:

$\{\mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{40})^{\mathsf{H}}, \mathbf{t}(p_{41})^{\mathsf{H}}\} \succ \{\mathbf{t}(p_{37})^{\mathsf{H}}, \mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{39})^{\mathsf{H}}\}$
$\mathbf{sg}(\tilde{q}_5) \leftarrow \{$
$\{\mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{40})^{\mathsf{H}}, \mathbf{t}(p_{41})^{\mathsf{H}}\} \succ \{\mathbf{t}(p_{37})^{\mathsf{H}}, \mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{39})^{\mathsf{H}}\}$
$\}$

This direction of preference is problematic when the two alternatives are compared in terms of cost. It is reasonable to assume that setting up the intranet to handle the exchange of data between the mobile and web applications requires the purchasing of licenses to software that would manage that exchange, while there would be no such costs when Internet is used as the infrastructure for data exchange. The softgoal and the corresponding preference generated by that softgoal are:

$\mathbf{sg}(\tilde{q}_9$: Keep the cost of SNM low)
$\{\mathbf{t}(p_{37})^{\mathsf{H}}, \mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{39})^{\mathsf{H}}\} \succ \{\mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{40})^{\mathsf{H}}, \mathbf{t}(p_{41})^{\mathsf{H}}\}$
$\mathbf{sg}(\tilde{q}_9) \leftarrow \{$
$\{\mathbf{t}(p_{37})^{\mathsf{H}}, \mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{39})^{\mathsf{H}}\} \succ \{\mathbf{t}(p_{38})^{\mathsf{H}}, \mathbf{t}(p_{40})^{\mathsf{H}}, \mathbf{t}(p_{41})^{\mathsf{H}}\}$
$\}$

It can be easily verified that because of these preference relations, there is a qualitative tradeoff relation between the two options. Tolerating the conflict between the two options keeps both in the requirements database, and thereby allows further discussion of the tradeoff between the stakeholders, rather than its premature resolution.

In both options, the tasks are the responsibilities of the Software engineer role:

$(\mathsf{R}_8\mathbf{t}(p_{37})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_8\mathbf{t}(p_{38})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_8\mathbf{t}(p_{39})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_8\mathbf{t}(p_{40})^{\mathsf{H}})^{\mathsf{H}}$
$(\mathsf{R}_8\mathbf{t}(p_{41})^{\mathsf{H}})^{\mathsf{H}}$

Refining $\mathbf{g}(p_{14})$.

11 Conclusions

Techne was presented as a family of mathematical formalisms. Each one formalizes within a propositional framework various concepts and relations suggested and discussed separately in prior work. Techne shows how choices of concepts and relations to have in a language result in different requirements problem and solution concepts. Techne illustrates how various prior ideas can be fitted together within the same language. Figure 3 summarizes the types of information that each Techne formalism can represent, and distinguishes primitive concepts and relations from derived ones.

Four issues were not discussed in this paper: automated reasoning to find solutions in each Techne formalism, nesting of modalities (i.e., allowing, e.g., $\mathbf{g}(\mathbf{t}(p))$ to be an expression), which decision rules are relevant in Techne formalisms having the preference relation, and the visualization of requirements databases.

References

[1] J.-R. Abrial. *The B-book: assigning programs to meanings*. Cambridge Univ. Press, New York, NY, USA, 1996.

[2] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):pp. 510–530, 1985.

[3] Anonymous. Report of the Inquiry Into The London Ambulance Service. Technical report, The Communications Directorate, South West Thames Regional Authority, 1993.

[4] L. Baresi, L. Pasquale, and P. Spoletini. Fuzzy Goals for Requirements-driven Adaptation. In *IEEE Int. Req. Eng. Conf.*, 2010.

[5] D. Bäumer, W. R. Bischofberger, H. Lichter, and H. Züllighoven. User interface prototyping - concepts, tools, and experience. In *ICSE*, pages 532–541, 1996.

[6] D. Bjørner and C. B. Jones, editors. *The Vienna Development Method: The Meta-Language*, volume 61 of *Lecture Notes in Computer Science*. Springer, 1978.

[7] E. M. Clarke and J. M. Wing. Formal methods: state of the art and future directions. *ACM Comput. Surv.*, 28(4):626–643, 1996.

[8] D. Dalcher. Disaster in London: The LAS Case study. In *ECBS*, pages 41–52. IEEE Computer Society, 1999.

[9] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.

	Concepts		Relations		
	Primitive	**Derived**	**Primitive**	**Derived**	
T1	Goal	Alternative	Conjunction	Inference	Means-ends
	Domain assumption	Argument	Inference	Conflict	Type-A Conflict
	Task	Problem	Consequence	Operationalization	Type-B Conflict
		Solution	Inconsistency	Goal refinement	Type-C Conflict
				Task decomposition	
T2	Goal	Alternative	Conjunction	Inference	Type-C Conflict
	Domain assumption	Argument	Inference	Conflict	Role dependency
	Task	Problem	Consequence	Operationalization	Agent dependency
	Agent	Solution	Inconsistency	Goal refinement	Cond. responsibility
	Role		Responsibility	Task decomposition	Role commitment
			Ability	Means-ends	Agent commitment
			Occcupancy	Type-A Conflict	Delegation
			Commitment	Type-B Conflict	
T3	Goal	Alternative	Conjunction	Inference	Agent dependency
	Domain assumption	Argument	Inference	Conflict	Cond. responsibility
	Task	Problem	Consequence	Operationalization	Role commitment
	Agent	Solution	Inconsistency	Goal refinement	Agent commitment
	Role	Decision rule	Responsibility	Task decomposition	Delegation
	Softgoal		Ability	Means-ends	Contribution
	Soft dom. assumption		Occcupancy	Type-A Conflict	Qual. relaxation
			Commitment	Type-B Conflict	Quant. relaxation
			Preference	Type-C Conflict	Labeled preference
				Role dependency	
T4	Goal	Alternative	Conjunction	Inference	Role commitment
	Domain assumption	Argument	Inference	Conflict	Agent commitment
	Task	Problem	Consequence	Operationalization	Delegation
	Agent	Threshold solution	Inconsistency	Goal refinement	Contribution
	Role	Candidate solution	Responsibility	Task decomposition	Qual. relaxation
	Softgoal	Solution	Ability	Means-ends	Quant. relaxation
	Soft dom. assumption	Decision rule	Occcupancy	Type-A Conflict	Labeled preference
			Commitment	Type-B Conflict	Strict inference
			Preference	Type-C Conflict	Defeasible inference
			Mandatory	Role dependency	Strict conflict
			Preferred	Agent dependency	Defeasible conflict
			Inherited	Cond. responsibility	
T5	Goal	Alternative	Conjunction	Inference	Role commitment
	Domain assumption	Argument	Inference	Conflict	Agent commitment
	Task	Problem	Consequence	Operationalization	Delegation
	Agent	Threshold solution	Inconsistency	Goal refinement	Contribution
	Role	Candidate solution	Responsibility	Task decomposition	Qual. relaxation
	Softgoal	Solution	Ability	Means-ends	Quant. relaxation
	Soft dom. assumption	Decision rule	Occcupancy	Type-A Conflict	Labeled preference
	Quality constraint		Commitment	Type-B Conflict	Strict inference
			Preference	Type-C Conflict	Defeasible inference
			Mandatory	Role dependency	Strict conflict
			Preferred	Agent dependency	Defeasible conflict
			Inherited	Cond. responsibility	Quantitative conflict

Figure 3: Overview of the primitive and derived concepts and relations in Techne formalisms.

[10] R. Darimont and A. van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *SIGSOFT FSE*, 1996.

[11] A. Fuxman, L. Liu, J. Mylopoulos, M. Roveri, and P. Traverso. Specifying and analyzing early requirements in Tropos. *Requirements Eng.*, 9(2):132–150, 2004.

[12] S. Greenspan, J. Mylopoulos, and A. Borgida. On formal requirements modeling languages: RML revisited. In *Proc. 16th Int. Conf. Software Eng.*, pages 135–147, 1994.

[13] S. J. Greenspan, A. Borgida, and J. Mylopoulos. A requirements modeling language and its logic. *Inf. Syst.*, 11(1):9–23, 1986.

[14] J. V. Guttag, J. J. Horning, S. J. Garland, K. D. Jones, A. Modet, and J.M. Wing. *Larch: languages and tools for formal specification*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[15] A. Hunter. Paraconsistent Logics. In D. M. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Reasoning with actual and potential contradictions*. Springer, 1998.

[16] D. Hyde. Sorites paradox. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[17] S. Jones, D. Till, and A. M. Wrightson. Formal methods and requirements engineering: Challenges and synergies. *Journal of Systems and Software*, 40(3):263–273, 1998.

[18] C. Kennedy. Vagueness and grammar: the semantics of relative and absolute gradable adjectives. *Linguistics and Philosophy*, 30(1):1–45, 2007.

[19] S. King, J. Hammond, R. Chapman, and A. Pryor. Is proof more cost-effective than testing? *IEEE Trans. Software Eng.*, 26(8):675–686, 2000.

[20] E. Letier and A. van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *SIGSOFT FSE*, pages 53–62, 2004.

[21] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. MIT Press, 2001.

[22] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: representing knowledge about information systems. *ACM Trans. Inf. Syst.*, 8(4):325–362, 1990.

[23] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Trans. Softw. Eng.*, 18(6):483–497, 1992.

[24] C. J. Neill and P. A. Laplante. Requirements engineering: The state of the practice. *IEEE Software*, 20(6):40–45, 2003.

[25] R. Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132, 1980.

[26] W. N. Robinson, S. D. Pawlowski, and V. Volkov. Requirements interaction management. *ACM Comput. Surv.*, 35(2):132–190, 2003.

[27] S. Shapiro. Classical logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2009 edition, 2009.

[28] G. R. Simari and R. P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell.*, 53(2-3):125–157, 1992.

[29] Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

[30] J. M. Spivey. *Introducing Z: A Specification Language and Its Formal Semantics*. Cambridge Univ. Press, 1988.

[31] D. C. Stidolph and E. J. Whitehead Jr. Managerial issues for the consideration and use of formal methods. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *FME*, volume 2805 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2003.

[32] A. van Lamsweerde, R. Darimont, and E. Letier. Managing conflicts in goal-driven requirements engineering. *IEEE Trans. Software Eng.*, 24(11):908–926, 1998.

[33] A. van Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Trans. Software Eng.*, 26(10):978–1005, 2000.

[34] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel. RELAX: a language to address uncertainty in self-adaptive systems requirements. *Requir. Eng.*, 15(2):177–196, 2010.

[35] J. M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8–24, 1990.

[36] J. Woodcock, P. Gorm Larsen, J. Bicarregui, and J. S. Fitzgerald. Formal methods: Practice and experience. *ACM Comput. Surv.*, 41(4), 2009.

[37] E. S. K. Yu and J. Mylopoulos. Understanding "Why" in Software Process Modelling, Analysis, and Design. In *Proc. 16th Int. Conf. Software Eng.*, pages 159–168, 1994.

[38] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM T. Softw. Eng. Methodol.*, 6(1):1–30, 1997.