How to Make Requirements Modelling Languages? An Introductory Tutorial

Ivan Jureta

Fonds de la Recherche Scientifique – FNRS
and Department of Business Administration, University of Namur
ivan.jureta@unamur.be

33rd International Conference on Conceptual Modelling (ER2014), Atlanta
Last update: October 26, 2014

Acknowledgments

I am grateful most of all to **John Mylopoulos** and **Alexander Borgida**, who were very patient in many discussions that we have had on these topics since 2007.

I have co-authored papers on RMLs with them, as well as with **Neil Ernst, Alberto Siena, Anna Perini, Angelo Susi, Stéphane Faulkner, Pierre-Yves Schobbens, and many others**. They have all influenced the content of this tutorial in one way or another.

This does not mean that we agree on the ideas which I present here.

Purpose

This tutorial is relevant if you need to solve **Requirements Problems repeatedly** and **in collaboration with others**.

You have a **Requirements Problem** to solve if:

1. there is partial information about expectations,
2. expectations are presently not satisfied,
3. a system should be made to satisfy them.

To solve Requirements Problems, you need rules on:

- How to elicit information about the problem?
- How to document it?
- How to organise it?
- How to draw conclusions from it?
- And do anything else that's needed to design a **Solution**.

Examples of Requirements Problems

London Ambulance Service:

Improve response to emergency incidents through new hardware, software, training


SAMSUNG
Samsung:
Sell more
through better merchandising
enabled by new hardware and software


Several startups:
Offer everyone better running training
through new hardware and software

What is common to these problems?

- No off-the-shelf solution,
- Solution must be designed, then produced,
- Design involves problem-solving and decision-making,
- Design requires different kinds of expert knowledge,
- Design takes time,
- Design output is used as an input to produce the solution,
- Design and production are done by different people.

What are the goals of this tutorial?

- Illustrate how to make new RMLs,
- Discuss some major topics in RML design,
- Briefly discuss designs of well-known RMLs.

Outline

1. General background (15 min)
2. Relations (50 min)
3. Guidelines (30 min)
4. Categories (25 min)
5. Alternatives (15 min)
6. Formal theories (20 min)
7. Designs of well-known RMLs (10 min)
8. Closing comments
9. Questions & answers

Tutorial material

You can go to

`http://www.jureta.net/requirements-modelling-languages`

and download these slides and the text which covers these and other topics in RML design, including:

- Valuation
- Uncertainty and Probability
- Preferences
- Problem Classes

General background

What is Requirements Engineering (RE)?

RE as a field of research involves:

- Descriptive research: Which RPs do people solve and how?
- Prescriptive research: How to better solve RPs?

There is research on such topics as:

- Elicitation,
- Modelling,
- Verification,
- Negotiation,
- Validation,
- Traceability,
- And so on.

What is the Default RP?

Default RP is:

Given:

- a set R of requirements, and
- a set K of domain knowledge,

Define a set S called "specification", which satisfies

- the provability condition $K, S \vdash R$, and
- the consistency condition $K, S \nvdash \bot$.

Default RP is due to Pamela Zave and Michael Jackson, in "Four dark corners of requirements engineering", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(1):1-30, 1997.

What does the provability condition say?

$$K, S \vdash R$$

There has to exist a proof of requirements, from domain knowledge and specifications.

What does the consistency condition say?

$$K, S \nvdash \bot$$

Information about the operating environment (domain knowledge) and expected system operation (specification) need to be logically consistent.

Why?

How do you get information in K, R, S?

Iteratively, starting from partial, unclear, imprecise, inconsistent information about stakeholders' expectations.

What role do Requirements Models play in RE?

Requirements Models are **external representations** in RP-solving.

External representations influence how people **discover, describe, and explore problems and their solutions**.

The above is supported by empirical evidence in cognitive science research, **but be careful**: that research was **not** done specifically for Requirements Models.

What role do RMLs play in RE?

RMLs directly or indirectly influence:

- Which information to elicit about a problem?
- How to represent that information?
- How to communicate that information to others?
- How to transform that information?
- What problem to solve?
- What is a solution to the problem?
- How quickly is the solution designed?
- How good the solution is?

Relations

Exercise 1: Incremental increase of problem information

Define the simplest RML which lets you show that information about the problem increases incrementally as system design progresses. By simplest, I mean something that is easy for others to understand.

Why is such a language interesting?

Because in problem-solving, there are:

- **Discovery**, which is to start with little and partial information, and progressively increase and improve it,
- **Indecision**, which is to postpone decisions until later in design.

How would you start to make this language?

You know at least the following:

- There can be all kinds of information about the problem,
- You want to use models with others,
- Models should not be (too) confusing to others,
- Models should let you show increase in information.

How would I start to make this language?

Define **one question**,

which **any** model of this language can answer,

and its answer would be **the same for any user** of that model.

Any such question is called a **Language Service**.

Why are Language Services interesting?

- Emphasise that language is a problem-solving tool,
- Describe what the language can do for you,
- Compare languages by the Language Services they deliver.

Which Language Service should this language deliver?

Recall:

Models should show **incremental increase of information** about the problem.

Suppose:

- Yesterday, you had a piece of information x in the model,
- Today, you found out y, which tells you a bit more about x.

How would you relate x and y, to show that one adds information to the other?

Which Language Service should this language deliver?

How about:

Language Service

AddsDetails: **Does *x* add information to *y* in *M*?**

How can the language deliver this Language Service?

In “*x* add information to *y*”:

- What are *x* and *y*?
- What is “adds information to”, or shorter “informs”?
- What are the properties of “informs”?
- What is an easy visualisation of “*x* informs *y*”?

How can the language deliver this Language Service?

In “*x* informs *y*”:

- What are *x* and *y*? **Fragments**.
- What is “informs”? **Binary relation**.
- What are the properties of “informs”?
  - **Irreflexive,**
  - **Antisymmetric,**
  - **Transitive.**
- What is an easy visualisation of “*x* informs *y*”? **Directed graph.**

A trivial modelling language, called L.D1

Every model M in L.D1 is a graph $(X, \text{r.inform})$, where:

1. every Fragment in X is a node,
2. every edge is an instance of **r.inform** over X,
3. **r.inform** is a strict partial order on members of X, and
4. $(x, y) \in$ r.inform reads "Fragment x adds details to Fragment y".

Does L.D1 deliver the Language Service?

Does *x* add information to *y* in model M?

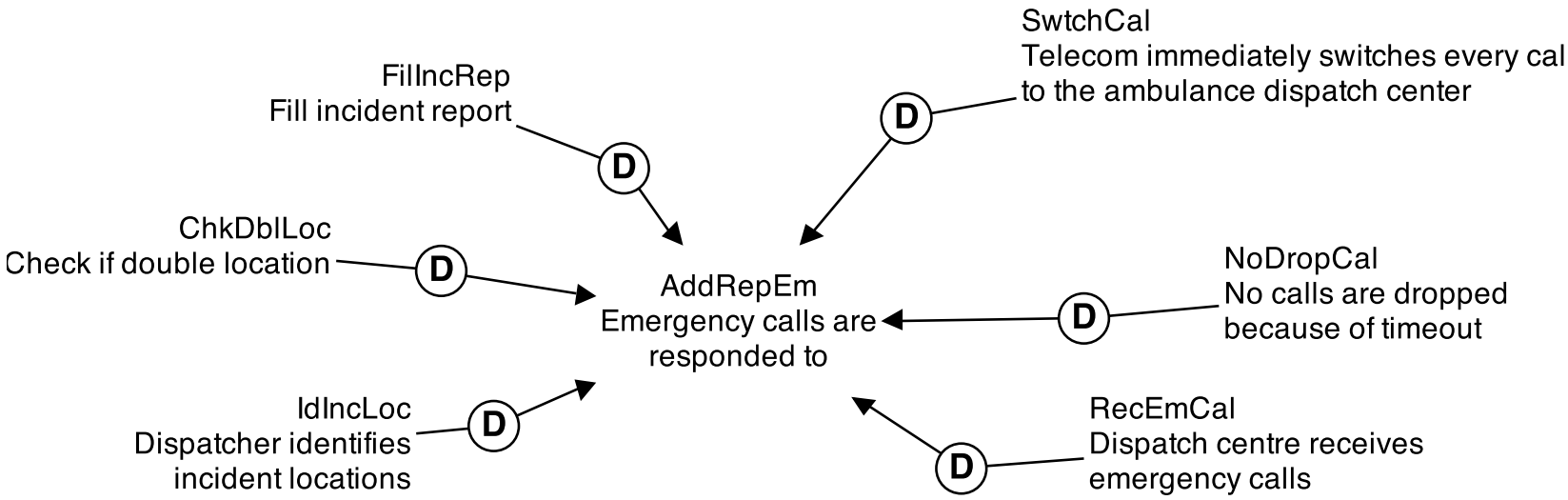
Yes, iff there is a path from *x* to *y* in the transitive closure of *M*.

Example

Suppose you have these Fragments:

- Emergency calls are responded to. (AddRepEm)
- Receive emergency calls. (RecEmCal)
- Switch emergency calls to smb. dispatch centre. (SwtchCal)
- No calls are dropped because of timeout. (NoDropCal)
- Identify the incident location. (IdIncLoc)
- Check if double location. (ChkDblLoc)
- Fill out the incident report. (FillIncRep)
- Fill out incident report form via software. (FillSwIncRep)

Example



A visualisation of a model made with L.D1.

Exercise 2: Define a relation as a reusable module

Define the relation **r.inform** so that it is independent of the syntax of **L.D1**, and that it can be reused when defining another language.

This other language may have an entirely different syntax, symbolic, visual, or otherwise.

Where to start?

Consider the following questions:

- What do you need to know about a relation, to use it in modelling?
- What does a relation definition include?
- When defining a relation, is it necessary to define how to represent its instances?
- What can be omitted from a definition of a relation?

What to know, when using a relation?

- Why do you need it?
- Its name?
- How to read its instances?
- Domain?
- Dimension (arity)?
- Properties to check models for?

I want to answer these questions for every relation, so I'll define a **Language Module**.

Language Module

Relation

Name? (r. **Abbreviation?**)

Domain & Dimension

r. **Abbreviation?** $\subseteq$ **Domain?**, where Domain elements are?

Properties

Properties to check models for?

Reading

How to read it?

Language Services

Language Services and how to deliver them?

Inform relation Language Module

Relation

Inform (r. **ifm**)

Domain & Dimension

r. **ifm** $\subseteq$ F $\times$ F, where F is a set of Fragments.

Properties

irreflexive, antisymmetric, and transitive.

Reading

$(x, y) \in$ r.ifm **reads "x adds information to y".**

Language Services

- **s.AddsDetails: Yes, if $(x, y) \in$ r.ifm is in M.**

Exercise 3 : Define the language which has only r.ifm

Define the simplest language whose models can represent instances of **r.ifm**.

The language definition should not redefine, or repeat the properties of **r.ifm**.

The name of this language is **L.Alpheratz**.

Where to start?

- What does the language have, that the relation does not?
- Where does the relation go in the language?
- What about syntax and semantics?

What is syntax in a RML?

As usual: symbols + rules for combining symbols.

In **L.Alpheratz**:

- What can the simplest symbols be?
- Which symbol combinations do you need?
- What rules do you use to combine symbols?

Perhaps syntax is not the best place to start

What if you start from the domain of the language?

What is the domain of a language?

What is the domain of L.Alpheratz?

The domain is made of “things” that the language talks about.

In L.Alpheratz, the domain includes:

- Fragments, and
- Instances of r.ifm.

Back to syntax

Syntax should have symbols for all there is in the domain.

Therefore, in **L.Alpheratz**, any model is a set of symbols, where some denote Fragments, others **r.ifm** instances.

How to define syntax?

As BNF rules, which gives the following “symbolic syntax”:

A model M in **L.Alpheratz** is a set of symbols

$$M = \{\phi_1, \ldots, \phi_n\},$$

where every ϕ is generated according to the following BNF rules:

$$\begin{aligned} \alpha &::= x \mid y \mid z \mid \ldots \\ \beta &::= (\alpha, \alpha) \\ \phi &::= \alpha \mid \beta \end{aligned}$$

Another syntax, also for L.Alpheratz

A model M in **L.Alpheratz** is a set of symbols

$$M = \{\phi_1, \ldots, \phi_n\},$$

where every ϕ is generated according to the following BNF rules:

$$\begin{array}{lcl} \alpha & ::= & x \mid y \mid z \mid \ldots \\ \beta & ::= & \alpha \xrightarrow{ifm} \alpha \\ \phi & ::= & \alpha \mid \beta \end{array}$$

I have no suggestion on which syntax to choose. It's up to you.

There is syntax, and domain, and what else?

The mapping from syntax to the domain.

In L.Alpheratz:

- α symbols map to Fragments,

$$\mathcal{D}(\alpha) \in \mathsf{F}$$

- β symbols map to r.ifm instances,

$$\mathcal{D}(\beta) \in \text{r.ifm}$$

$\mathcal{D}$ denotes the function that maps syntax to domain.

Language: Alpheratz

Alpheratz

Language Modules

F, r.ifm

Domain

Set F of Fragments and r.ifm $\subseteq$ F $\times$ F.

Syntax

A model *M* in the language is a set of symbols $M = \{\phi_1, \ldots, \phi_n\}$, where every ϕ is generated according to the following BNF rules:

$$\begin{aligned} \alpha &::= x \mid y \mid z \mid \ldots \\ \beta &::= (\alpha, \alpha) \\ \phi &::= \alpha \mid \beta \end{aligned}$$

Mapping

$\mathcal{D}(\alpha) \in F$ and $\mathcal{D}(\beta) \in$ r.ifm, that is, every α represents a Fragment from F and every β an instance of r.ifm.

Language Services

Same as r.ifm.

Exercise 4: Find most and least detailed Fragments

How would you change L.Alpheratz, to deliver these Language Service:

- **s.MostDetails**: Which Fragments in *M* are **the most detailed**?
- **s.LeastDetails**: Which Fragments in *M* are **the least detailed**?

What can you map a strict partial order to?

A directed graph.

Hence two questions:

- How would this help you deliver **s.MostDetails** and **s.LeastDetails**?
- How could you “add” this ability to map to directed graphs, to **L.Alpheratz**?

How about a function?

The function would take a model in L.Alpheratz, and return a directed graph.

The function would deliver this Language Service:

Language Service
RelGraph: What graph is induced by the relation r.R over Fragments in F?

Language Module for the function

Function

Map a binary relation to a graph
(f.map.abrel.g)

Input

Set F of Fragments and a binary antisymmetric relation r.R $\subseteq$ F $\times$ F.

Do

Let G(F, r.R) $= (N, E, l_N, l_E)$ be an empty labelled directed graph. For every Fragment $f_i \in$ F, add a node n_i to N and let the Fragment label the node, $l_N(n_i) = f_i$. For every relation instance $(f_i, f_j) \in$ r.R, add an edge $(n_i, n_j) \in E$ to the graph, and label the edge r.R.

Output

G(F, r.R).

Language Services

- s.RelGraph: G(F, r.R).

Language Module for the function

Function

Map a binary relation to a graph
(f.map.abrel.g)

Input

Set F of Fragments and a binary antisymmetric relation r.R $\subseteq$ F $\times$ F.

Do

Let G(F, r.R) $= (N, E, l_N, l_E)$ be an empty labelled directed graph. For every Fragment $f_i \in$ F, add a node n_i to N and let the Fragment label the node, $l_N(n_i) = f_i$. For every relation instance $(f_i, f_j) \in$ r.R, add an edge $(n_i, n_j) \in E$ to the graph, and label the edge r.R.

Output

G(F, r.R).

Language Services

- s.RelGraph: G(F, r.R).

Language Module for the function

Function

Map a binary relation to a graph
(f.map.abrel.g)

Input

Set F of Fragments and a binary antisymmetric relation r.R $\subseteq$ F $\times$ F.

Do

Let G(F, r.R) $= (N, E, l_N, l_E)$ be an empty labelled directed graph. For every Fragment $f_i \in$ F, add a node n_i to N and let the Fragment label the node, $l_N(n_i) = f_i$. For every relation instance $(f_i, f_j) \in$ r.R, add an edge $(n_i, n_j) \in E$ to the graph, and label the edge r.R.

Output

G(F, r.R).

Language Services

- s.RelGraph: G(F, r.R).

Language Module for the function

Function

Map a binary relation to a graph
(f.map.abrel.g)

Input

Set F of Fragments and a binary antisymmetric relation r.R $\subseteq$ F $\times$ F.

Do

Let G(F, r.R) $= (N, E, l_N, l_E)$ be an empty labelled directed graph. For every Fragment $f_i \in$ F, add a node n_i to N and let the Fragment label the node, $l_N(n_i) = f_i$. For every relation instance $(f_i, f_j) \in$ r.R, add an edge $(n_i, n_j) \in E$ to the graph, and label the edge r.R.

Output

G(F, r.R).

Language Services

- **s.RelGraph: G(F, r.R).**

Back to s.LeastDetails and s.MostDetails

Add f.map.abrel.g to L.Alpheratz.

If $Cl(G(M))$ is the transitive closure of $G(M)$, where $G(M)$ is the graph produced by **f.map.abrel.g**, then:

- s.MostDetails: All nodes in $Cl(G(M))$ which have no incoming edges.
- s.LeastDetails: All nodes in $Cl(G(M))$ which have no outgoing edges.

Influence Relations

Exercise 5: Define one or more relations to represent influence

Define a relation which represents that the satisfaction of a Fragment depends on the satisfaction of another Fragment.

Some questions to start with

- How can a Fragment be influenced?
- What kinds of influence can there be between Fragments?
- If there are different kinds of influences, would you define a new relation for each?
- What if some influences are stronger than others?

How can a Fragment be influenced?

You can influence the property of a Fragment.

In RE, influence relations are about satisfaction.

If x influences y, then the satisfaction of y depends on that of x.

Start with the simplest influence relation

Consider this Language Service:

Language Service

DoesInfluence: **Does the satisfaction of x influence the satisfaction of y in model M?**

It does not matter if this influence is positive or negative, or is stronger or weaker than the influence of another Fragment.

Exercise 6: Define the relation which delivers s.DoesInfluence

Define a relation which conveys only that the satisfaction of a Fragment somehow influences the satisfaction of another Fragment. Whether this influence is positive or negative, or is stronger or weaker than the influence of another Fragment, is not relevant in this exercise.

How would you define the new Influence relation?

- Name?
- Abbreviation?
- Reading?
- Domain?
- Properties to check?
- Language Services?

How would you define the new Influence relation?

- Name? **Influence**
- Abbreviation? **inf**
- Reading? “**satisfaction of** *y* **depends on that of** *y*”
- Domain? **Fragments**
- Dimension? **Binary**
- Properties to check? **Irreflexive and transitive**
- Language Services? **s.DoesInfluence**

Language Module for r.inf

Relation

Influence (r.inf)

Domain & Dimension

r.inf $\subseteq$ F $\times$ F, where F is a set of Fragments.

Properties

irreflexive and transitive.

Reading

$(x, y) \in$ r.inf reads "the satisfaction of x influences the satisfaction of y", or equivalently, "there is a function $SatVal(y) = f(\ldots, SatVal(x))$".

Language Services

- s.DoesInfluence: Yes, if $(x, y) \in$ r.inf is in M.

And the language?

Language: Alpheratz2

Alpheratz2

Language Modules

F, **r.inf**

Domain

Set F of Fragments and **r.inf** $\subseteq$ **F** $\times$ **F** .

Syntax

Same as L.Alpheratz.

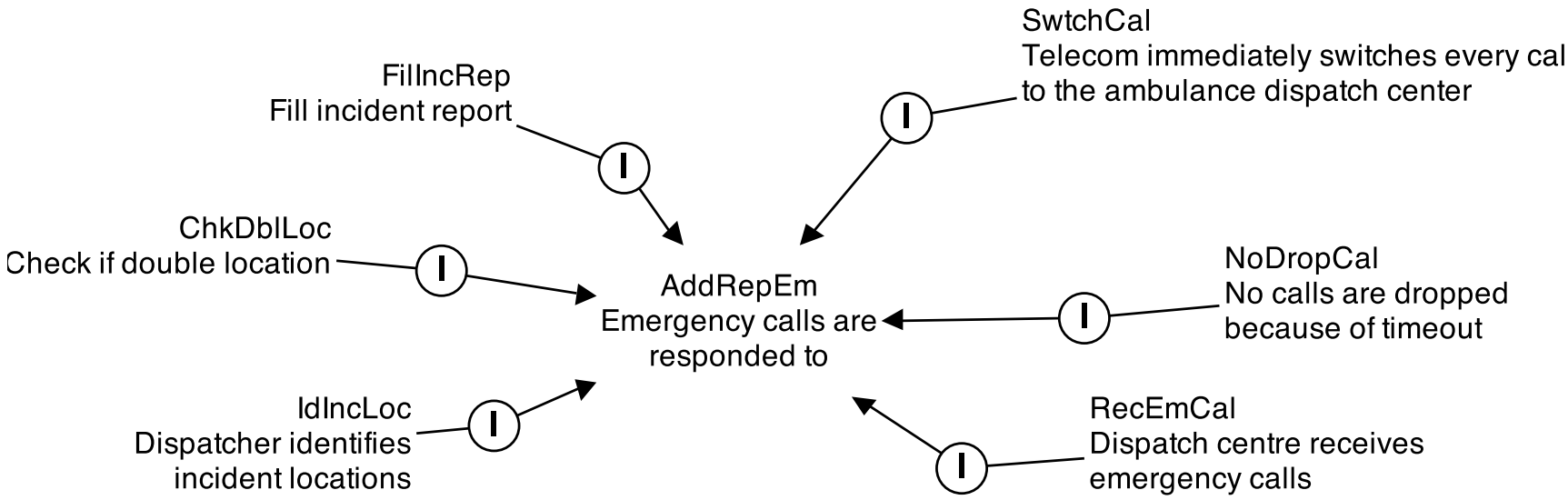
Mapping

$\mathcal{D}(\alpha) \in F$ and $\mathcal{D}(\beta) \in$ **r.inf** , that is, every α represents a Fragment from F and every β an instance of r.inf.

Language Services

Same as r.inf.

Example


FilIncRep
Fill incident report
SwtchCal
Telecom immediately switches every cal
to the ambulance dispatch center
ChkDblLoc
Check if double location
AddRepEm
Emergency calls are
responded to
NoDropCal
No calls are dropped
because of timeout
IdIncLoc
Dispatcher identifies
incident locations
RecEmCal
Dispatch centre receives
emergency calls

What if some influences are positive, and some negative?

That is, consider these Language Services:

Language Service

PosInfluence: Does satisfying x influences **positively** the satisfaction of y in M?

Language Service

NegInfluence: Does satisfying x influences **negatively** the satisfaction of y in M?

Exercise 7: Provide s.PosInfluence and s.NegInfluence

Define one or more relations which can provide s.PosInfluence and s.NegInfluence.

Questions to start with

- Do you need one relation, or more to deliver s.PosInfluence and s.NegInfluence?
- If more, then how are they different?
- Could you define one influence relation, whose parameter would define the direction of influence?

Positive influence

Relation

Influence.Positive (r.inf.pos)

Domain & Dimension

r.inf.pos $\subseteq$ F $\times$ F, where F is a set of Fragments.

Properties

irreflexive and transitive.

Reading

$(x, y) \in$ r.inf.pos reads “the satisfaction of x positively influences that of y”.

Language Services

- s.PosInfluence: Yes, if $(x, y) \in$ r.inf.pos is in M.

Negative influence

Relation

Influence.Negative (r.inf.neg)

Domain & Dimension

r.inf.neg $\subseteq$ F $\times$ F, where F is a set of Fragments.

Properties

irreflexive and transitive.

Reading

$(x, y) \in$ r.inf.neg reads “the satisfaction of x negatevely influences that of y”.

Language Services

- s.NegInfluence: Yes, if $(x, y) \in$ r.inf.neg is in M.

Influence with a parameter

Relation

Influence.d (r.inf. d)

Domain & Dimension

r.inf. **d** $\subseteq$ F $\times$ F, where F is a set of Fragments.

Properties

irreflexive and transitive.

Reading

d is either “pos” for positive or “neg” for negative, and therefore

- $(x, y) \in$ r.inf.pos reads “the satisfaction of x positively influences that of y”,
- $(x, y) \in$ r.inf.neg reads “the satisfaction of x negatevely influences that of y”.

Language Services

- s.PosInfluence: Yes, if $(x, y) \in$ r.inf.pos is in M.
- s.NegInfluence: Yes, if $(x, y) \in$ r.inf.neg is in M.

A language which delivers s.PosInfluence and s.NegInfluence

Language: Ankaa

Ankaa

Language Modules

F, r.inf.pos, r.inf.neg, f.map.abrel.g

Domain

Set F of Fragments. r.inf.pos and r.inf.neg are both over Fragments, so that r.inf.pos $\subseteq$ F $\times$ F and r.inf.neg $\subseteq$ F $\times$ F.

Syntax

Same as L.Alpheratz.

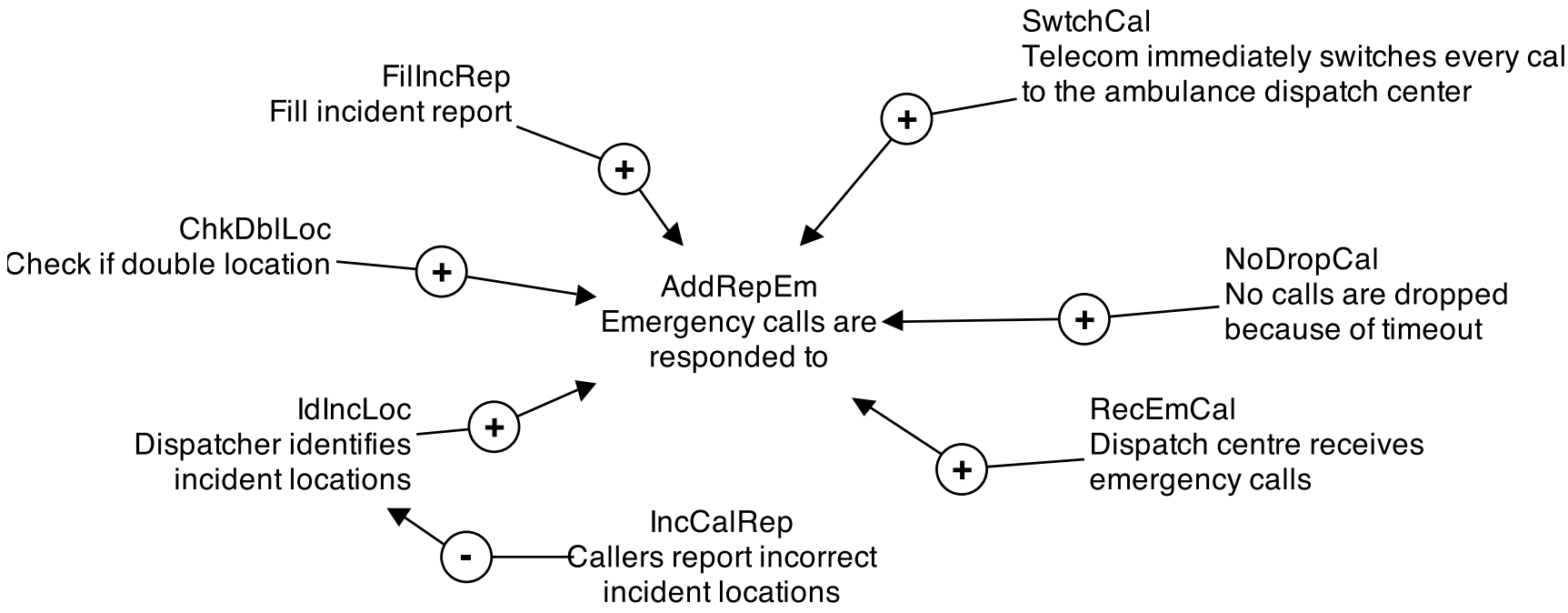
Mapping

α symbols denote Fragments, $\mathcal{D}(\alpha) \in$ F, β symbols denote r.inf.pos or r.inf.neg instances, $\mathcal{D}(\beta) \in$ r.inf.pos $\cup$ r.inf.neg.

Language Services

s.PosInfluence, s.NegInfluence.

Example



A visualisation of a model in L.Ankaa.

Exercise 8: Represent strength of influence between Fragments

Define a relation, or otherwise, which can be used to convey that the satisfaction of a Fragment more or less strangly influences that of another Fragment.

In other words, provide the following

Language Service

InfStrength:
In the model M, **if** the satisfaction of each of $x_1, \ldots, x_n$ influences the satisfaction of y,
then is the satisfaction of y more sensitive to the satisfaction of x_i than to the satisfaction of x_j,
where $x_i, x_j \in \{x_1, \ldots, x_n\}$?

Start with these questions

- Can influence strength be shown with a new relation?
- What is the scale for strength of influence?
- Is influence absolute or relative?
- If relative, then what is influence relative to?

How much do you need to know, to model influence strength?

- If absolute, then you need to know
  - A total order of strength values,
  - A function that maps every Fragment to a strength value,
- If relative, then you need to know less, namely
  - What strength is relative to,
  - A partial order of strength values,
  - Optionally, which strength values are equivalent.

“Stronger than” relation

Relation

Stronger influence (r.str.inf)

Domain & Dimension

r.str.inf $\subseteq R \times R$, where R is one of r.inf, r.inf.pos, r.inf.neg.

Properties

irreflexive, antisymmetric, and transitive.

Reading

$((x_i, y), (x_j, y)) \in$ r.str.inf reads “the satisfaction of y is more sensitive to the satisfaction of x_i than to the satisfaction of x_j”.

Language Services

- s.InfStrength: Yes, if $((x_i, y), (x_j, y)) \in$ r.str.inf is in M.

How to add r.infester to L.Alpheratz?

What has to change? How?

- Domain?
- Syntax?
- Mapping?
- Language Services?

A language with r.str.inf

Language

Schedar

Language Modules

F, r.inf.pos, r.inf.neg, f.map.abrel.g, r.str.inf

A language with r.str.inf

Language

Schedar

Domain

Set F of Fragments, r.inf.pos $\subseteq$ F $\times$ F, r.inf.neg $\subseteq$ F $\times$ F, and

$$\text{r.str.inf} \subseteq (\text{r.inf.pos} \times \text{r.inf.pos}) \cup (\text{r.inf.neg} \times \text{r.inf.neg}).$$

A language with r.str.inf

Language

Schedar

Syntax

A model M in the language is a set of symbols $M = \{\phi_1, \ldots, \phi_n\}$, where every ϕ is generated according to the following BNF rules:

$$
\begin{aligned}
\alpha &::= x \mid y \mid z \mid \ldots \\
\beta &::= (\alpha, \alpha) \\
\gamma &::= (\beta, \beta) \\
\phi &::= \alpha \mid \beta \mid \gamma
\end{aligned}
$$

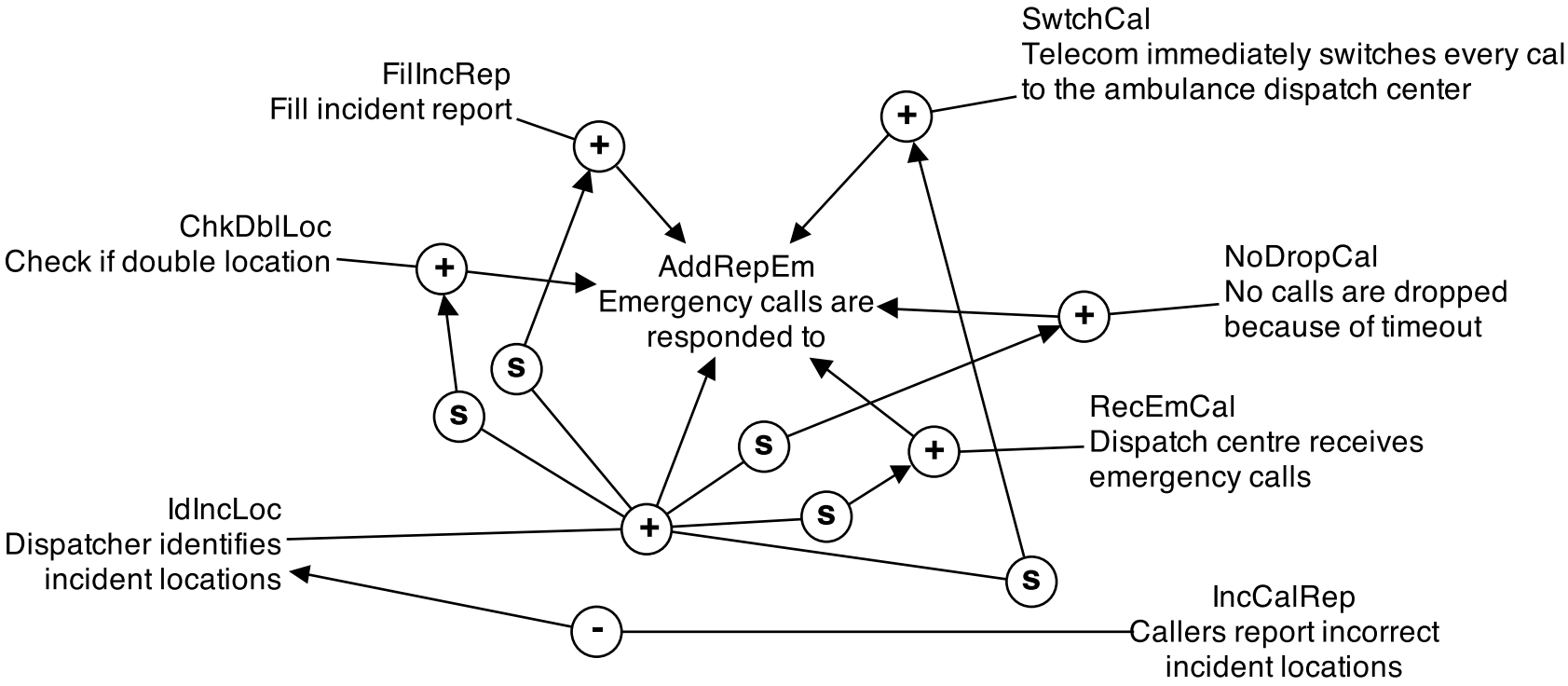
A language with r.str.inf

Language
Schedar
Mapping
$\mathcal{D}(\alpha) \in F$, $\mathcal{D}(\beta) \in$ r.inf.pos $\cup$ r.inf.neg, and $\mathcal{D}(\gamma) \in$ r.str.inf.

A language with r.str.inf

Language
Schedar
Language Services
s.PosInfluence, s.NegInfluence, s.InfStrength.

Example



A visualisation of a model in L.Schedar.

Is it interesting to have all influence relations in one language?

Yes, because each corresponds to a different level of knowledge about influence:

- There is influence, direction and strength unknown: use r.inf.
- Influence direction known, strength unknown: use r.inf.d.
- Influence direction and strength known: use r.inf.d and r.str.inf.

Argumentation / rationale relations

Exercise 9 : Show, in a model, information which justifies the content of that model

Define relations which can be used to show, in models, that some Fragments are arguments, reasons for having other parts of the model, such as other Fragments or relation instances.

Some questions to start with

- Can you do this with one relation?
- If not, then why?
- How would you use these relations to determine if a model part is justified or not?

Why is model rationale interesting in RE?

Suppose that a model includes this relation instance:

$$(x, y) \in \text{r.ifm}.$$

What if you are **not** convinced that x adds details to y?

Why accept that $(x, y) \in$ r.ifm in some M?

- Because of how r.ifm is defined?
- Because the person who added it to the model says so?
- Because of some properties of x and y?

Why accept that $(x, y) \in$ r.ifm in some M?

- Because of how r.ifm is defined?
  If yes, then what part of r.ifm definition is relevant?
- Because the person who added it to the model says so?
  If yes, then how to make sure everyone else accept?
- Because of some properties of x and y?
  If yes, then which properties? Where are they defined?

Why accept that $(x, y) \in$ r.ifm in some M?

Two approaches:

- Define observer-indepdnent properties that x and y should have, and which anyone can check in some standard way,
- Define a justification process, for determining if it is acceptable to say that x adds details to y.

Which relations do you need to do justification?

These relations should deliver:

Language Service

DoesSupport: Does accepting *x* support accepting *y* as well in *M*?

Language Service

DoesDefeat: Does accepting *x* support rejecting (not accepting) *y* in *M*?

A relation for s.DoesSupport

Relation

Support (r.sup)

Domain & Dimension

r.sup $\subseteq X \times X$, where X is either a set of Fragments or relation instances.

Properties

irreflexive, antisymmetric, and transitive.

Reading

$(x, y) \in$ r.Support reads "if x is accepted, then y should be".

Language Services

- s.DoesSupport: Yes, if there is $(x, y) \in$ r.Support in M.

A relation for s.DoesDefeat

Relation

Defeat (r.def)

Domain & Dimension

r.def $\subseteq X \times X$, where X is either a set of Fragments or relation instances.

Properties

irreflexive, antisymmetric, and intransitive.

Reading

$(x, y) \in$ r.Defeat reads “if x is accepted, then y should not be”.

Language Services

- s.DoesDefeat: Yes, if there is $(x, y) \in$ r.Defeat in M.

A language with r.sup and r.def

Language

Diphda

Language Modules

F, r.ifm, r.sup, r.def, f.map.abrel.g

A language with r.sup and r.def

Language

Diphda

Syntax

A model M in the language is a set of symbols $M = \{\phi_1, \ldots, \phi_n\}$, where every ϕ is generated according to the following BNF rules:

$$
\begin{aligned}
\alpha &::= x \mid y \mid z \mid \ldots \\
\beta &::= (\alpha, \alpha) \\
\gamma &::= (\alpha, \beta) \\
\phi &::= \alpha \mid \beta \mid \gamma
\end{aligned}
$$

A language with r.sup and r.def

Language

Diphda

Domain

F is a set of Fragments. r.ifm is over Fragments, so r.ifm $\subseteq$ F $\times$ F. A Fragment can act as a reason, or argument in favour or against a r.ifm instance or another Fragment, so that

$$\begin{aligned} \text{r.sup} &\subseteq (\text{F} \times \text{r.ifm}) \cup (\text{F} \times \text{F}), \\ \text{r.def} &\subseteq (\text{F} \times \text{r.ifm}) \cup (\text{F} \times \text{F}). \end{aligned}$$

A language with r.sup and r.def

Language

Diphda

Mapping

α symbols denote Fragments, so $\mathcal{D}(\alpha) \in$ F. β symbols denote r.ifm instances, or instances of r.sup or r.def between Fragments, that is,

$$\mathcal{D}(\beta) \in \text{r.inf} \cup \text{r.sup} \cup \text{r.def}.$$

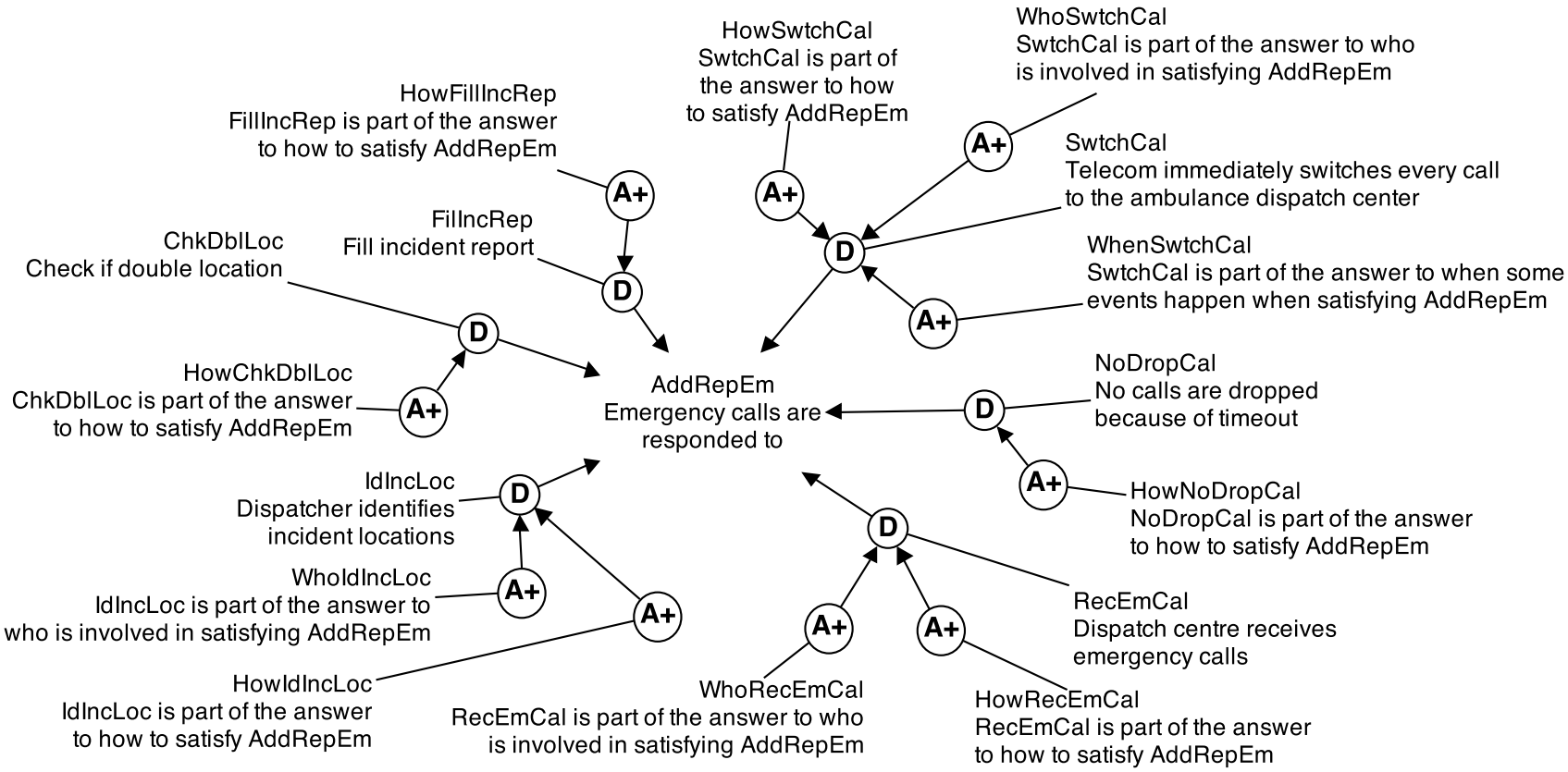
γ symbols denote r.sup or r.def from a Fragment to a r.ifm instance,

$$\mathcal{D}(\gamma) \in \text{r.sup} \cup \text{r.def}.$$

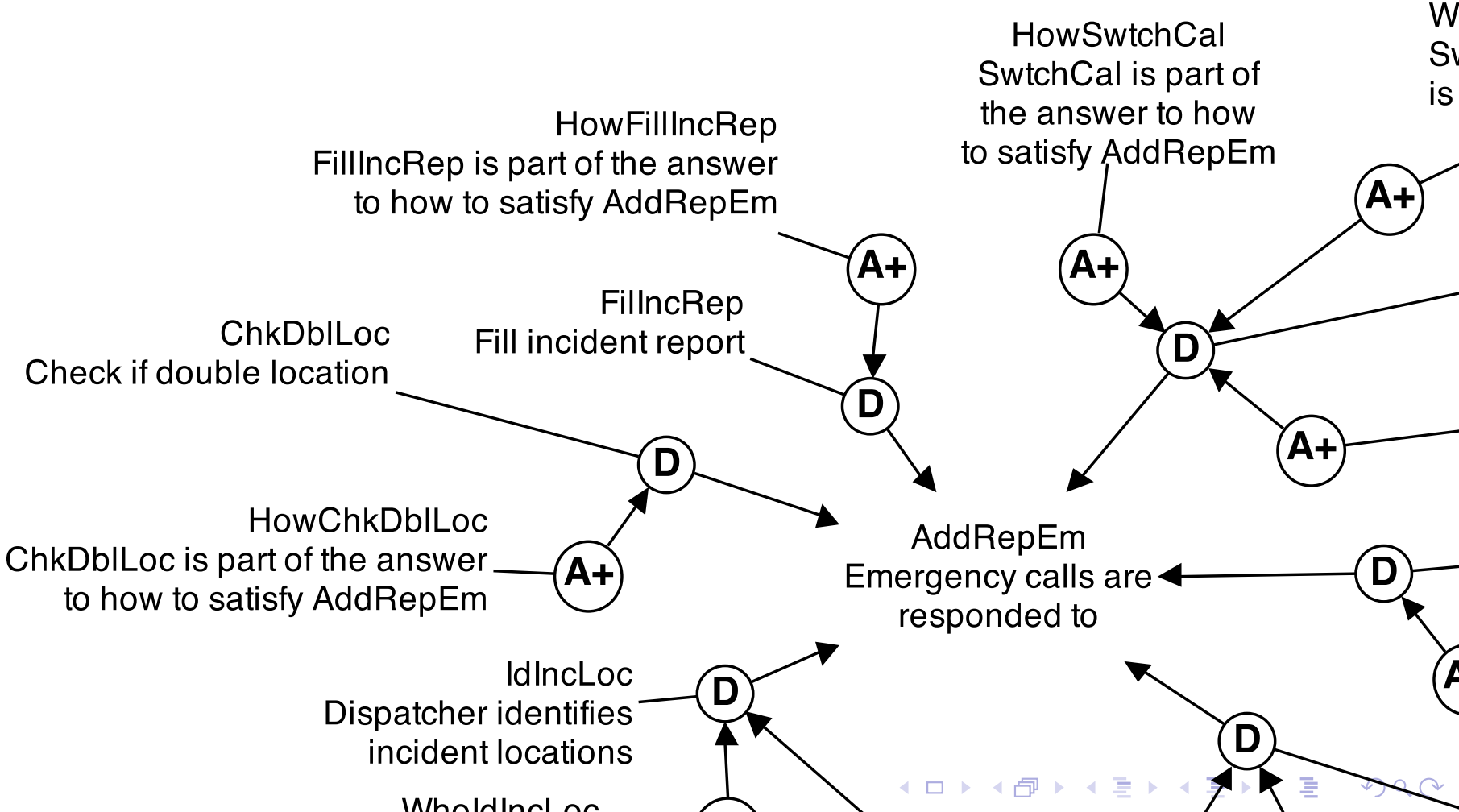
A language with r.sup and r.def

Language
Diphda
Language Services s.DoesSupport, s.DoesDefeat.

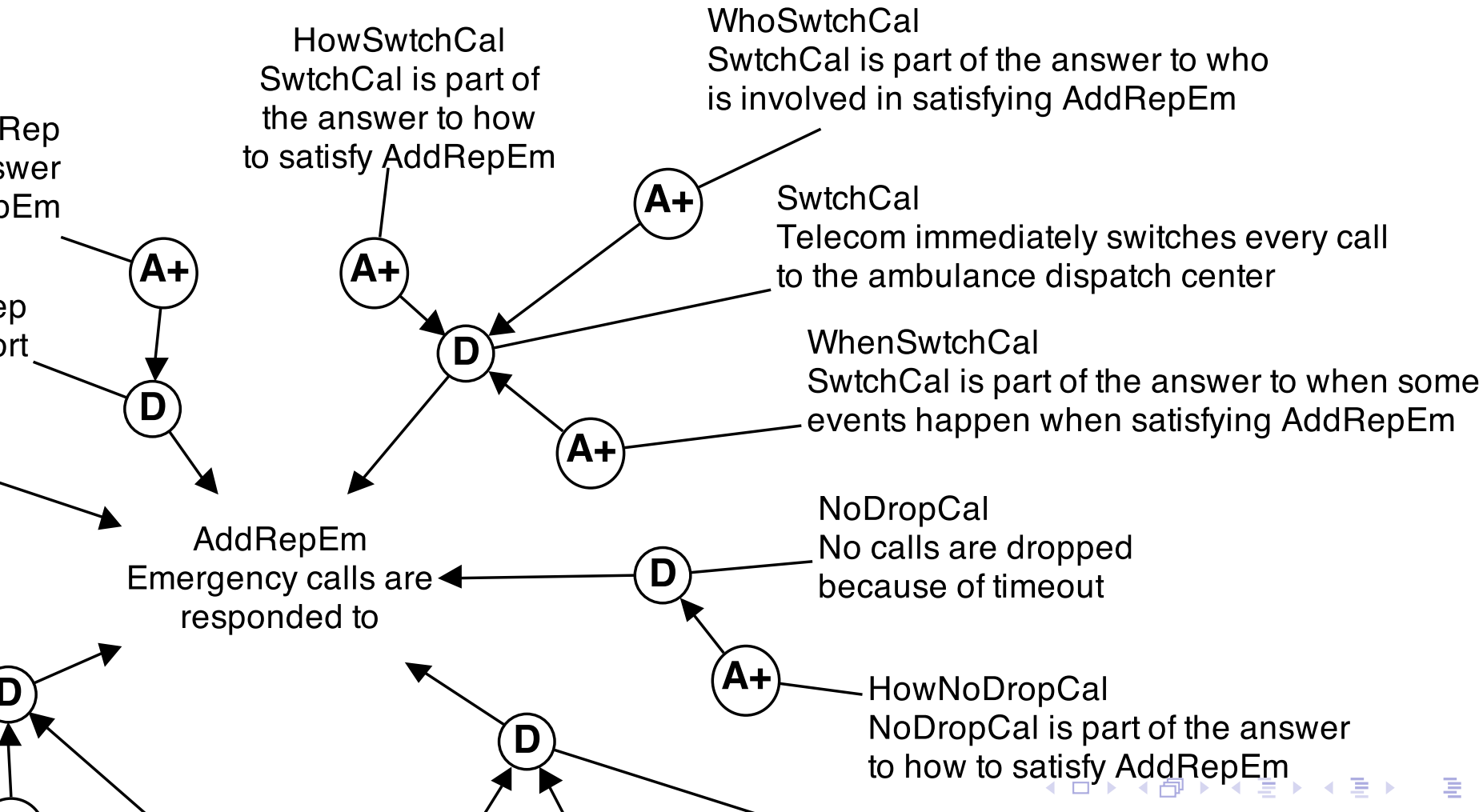
Example: A visualisation of a model in L.Diphda



Example: Zoom in



Example: Zoom in



Exercise 10: Define a procedure which computes if a model part is acceptable

L.Diphda can represent arguments for and against in models.

- Given a model which includes arguments, which arguments are acceptable (justified), and which are not?
- How can you compute this? How can the ability to compute this be built into a language?

The exercise is about how to deliver this Language Service

Language Service

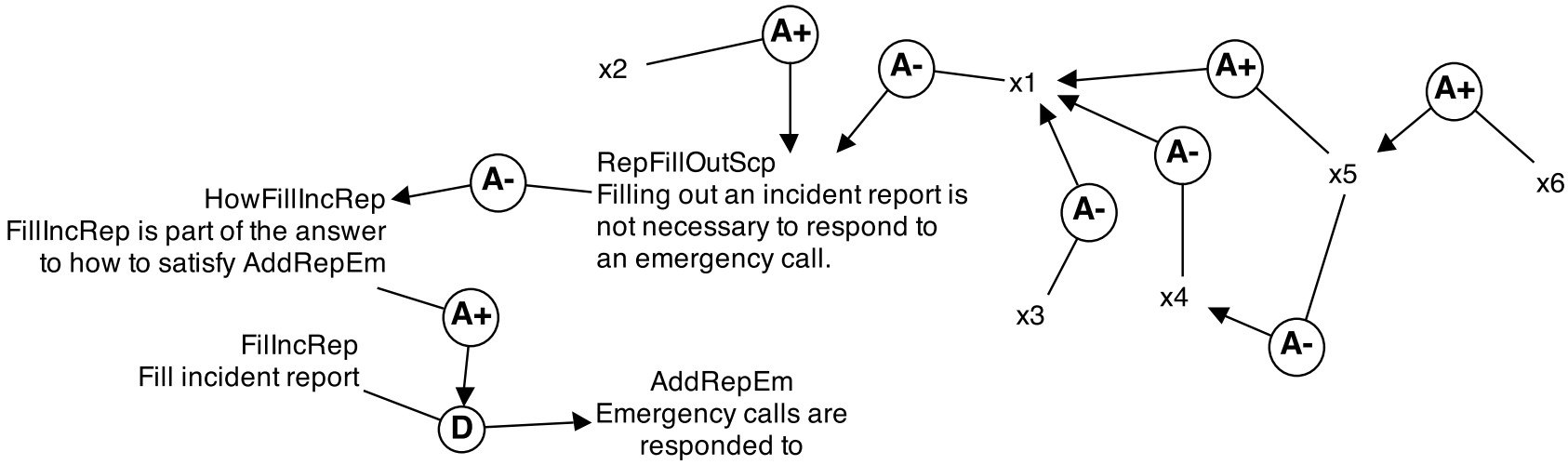
IsAcceptable: Is w acceptable in W, given relations r.sup and r.def over W?

How to deliver s.IsAcceptable?

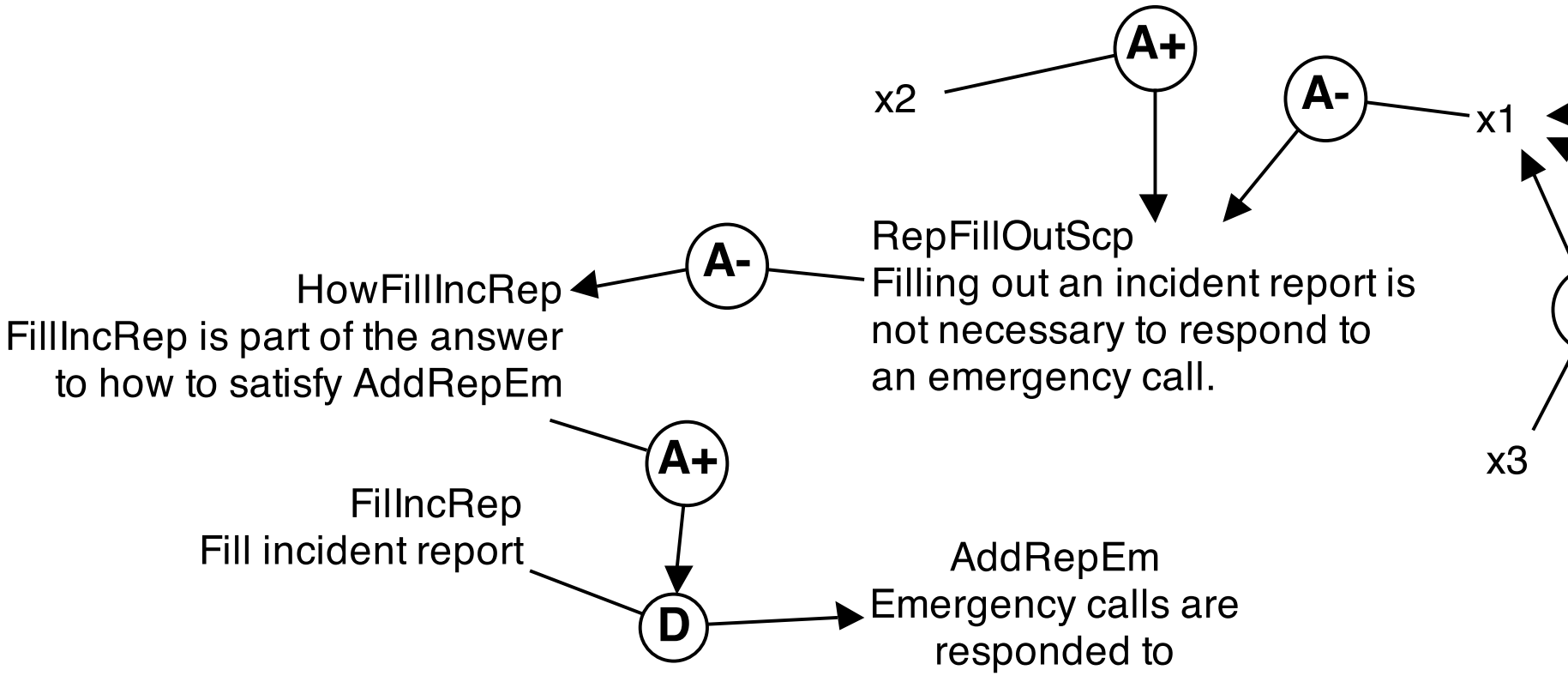
Suppose that you use the following rules:

1. Each Fragment or relation instance *w* gets 1 if acceptable, 0 if defeated (not accepted).
2. Leaves are acceptable.
3. Defeaters defeat.

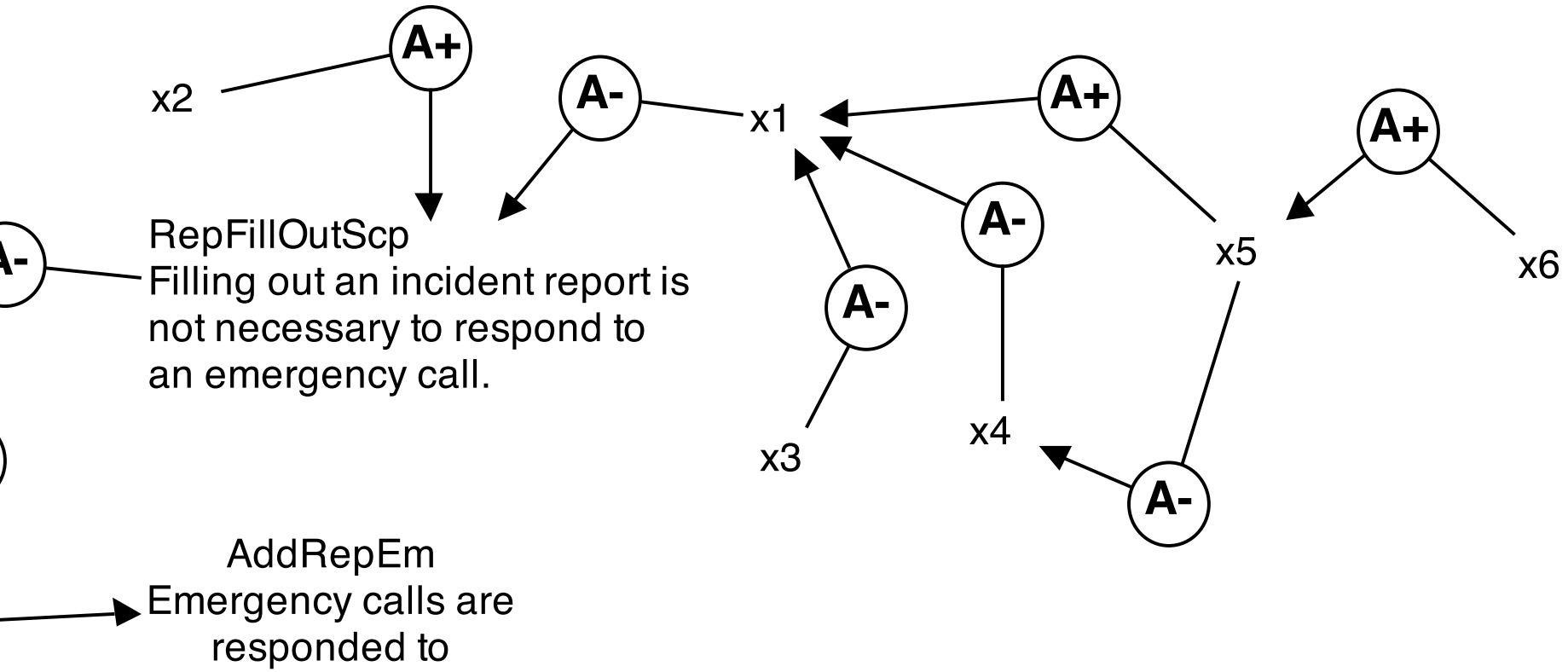
Example: A graph of support, defeat, and inform relations



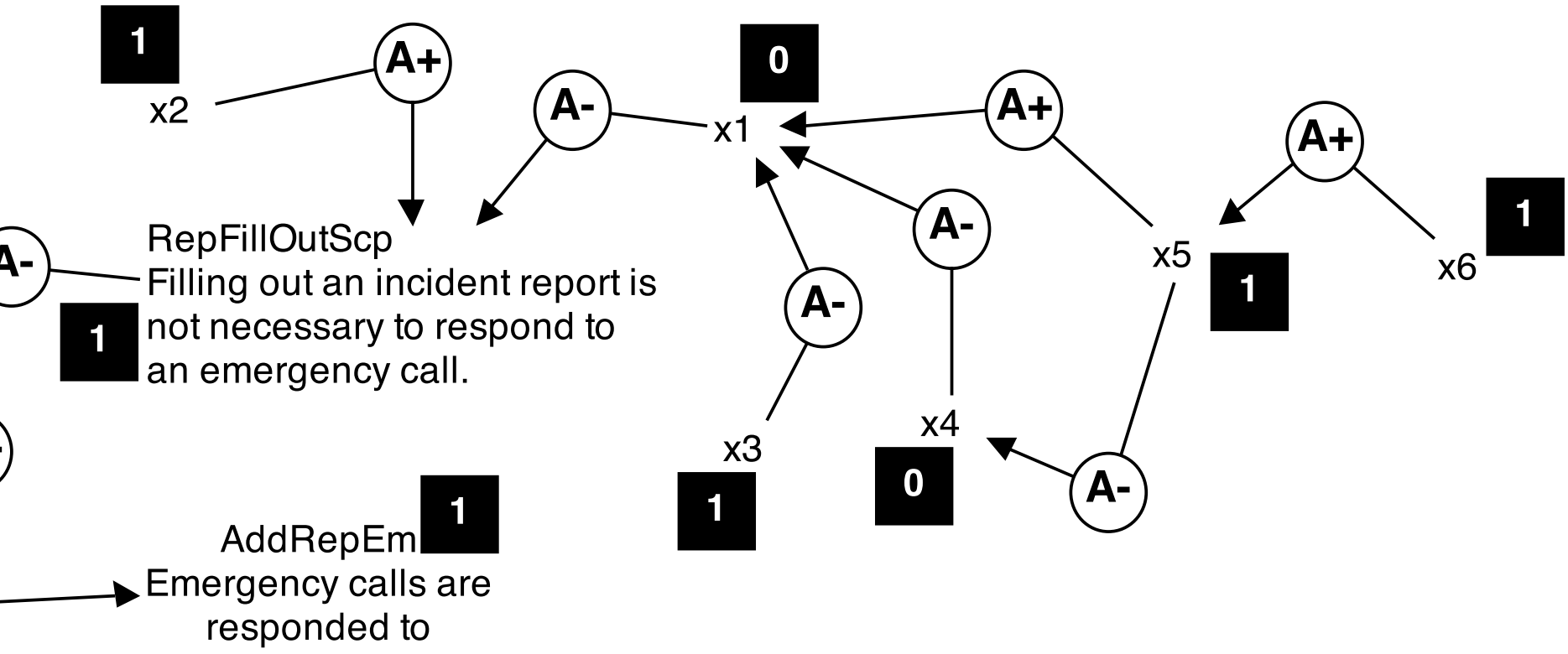
Example: Zoom in



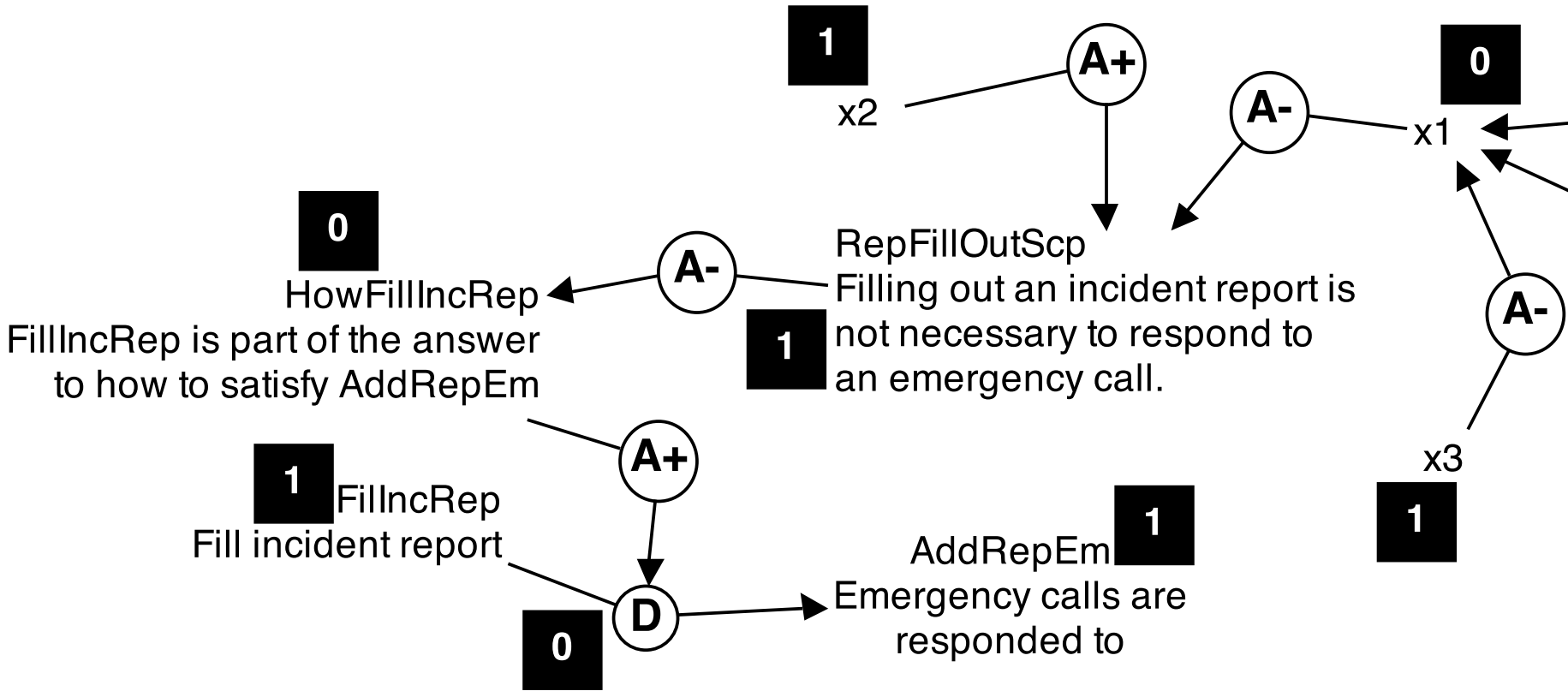
Example: Zoom in



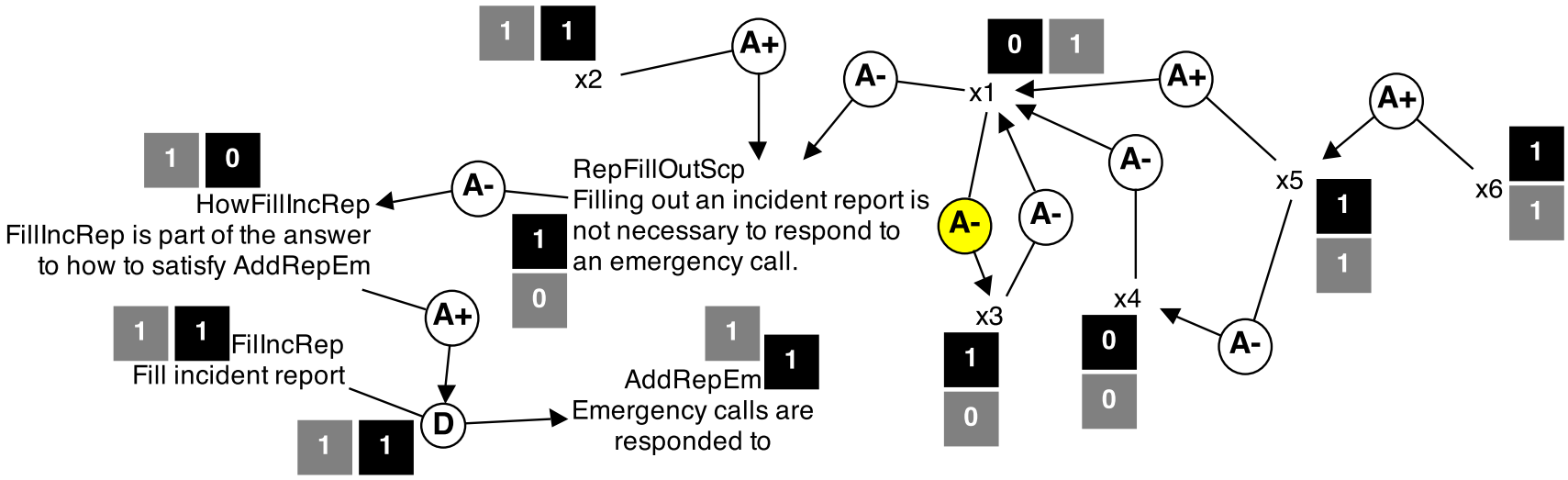
Example: Start assigning values from leaves



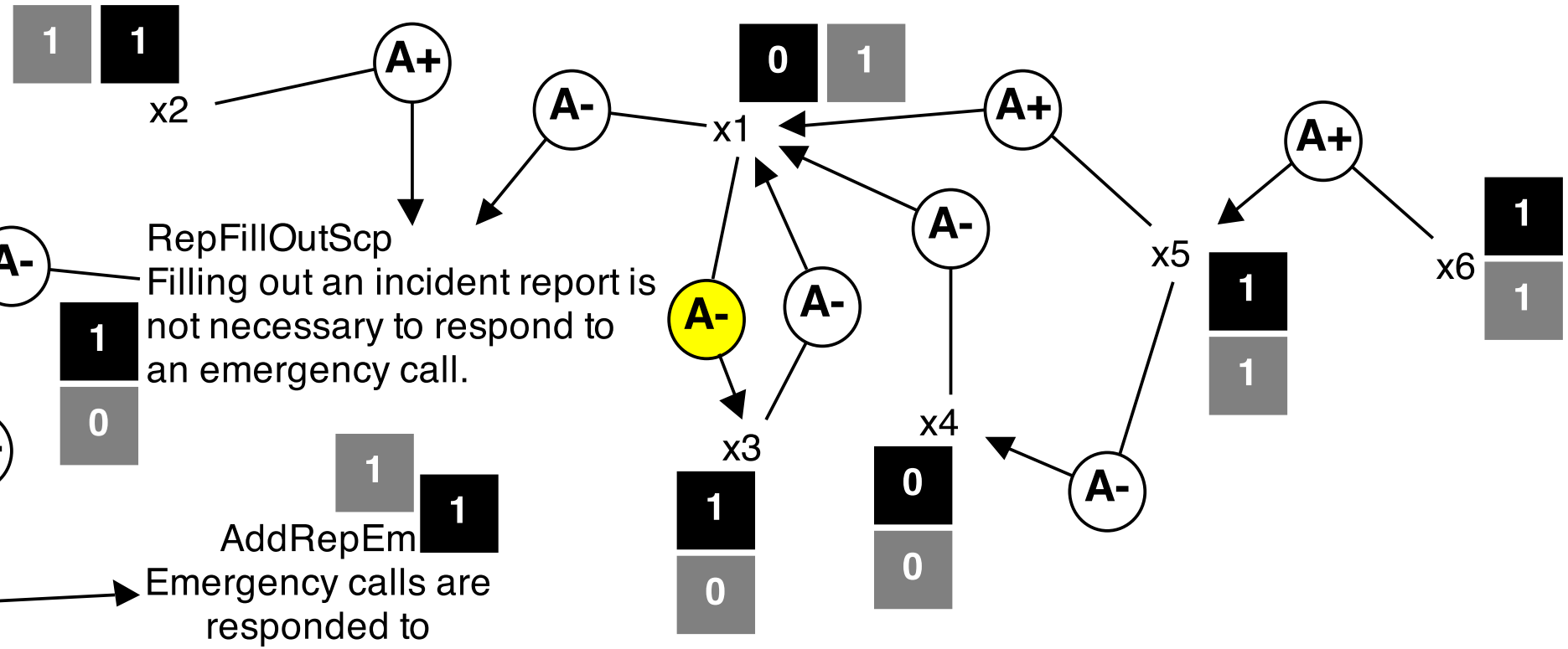
Example: Propagate values



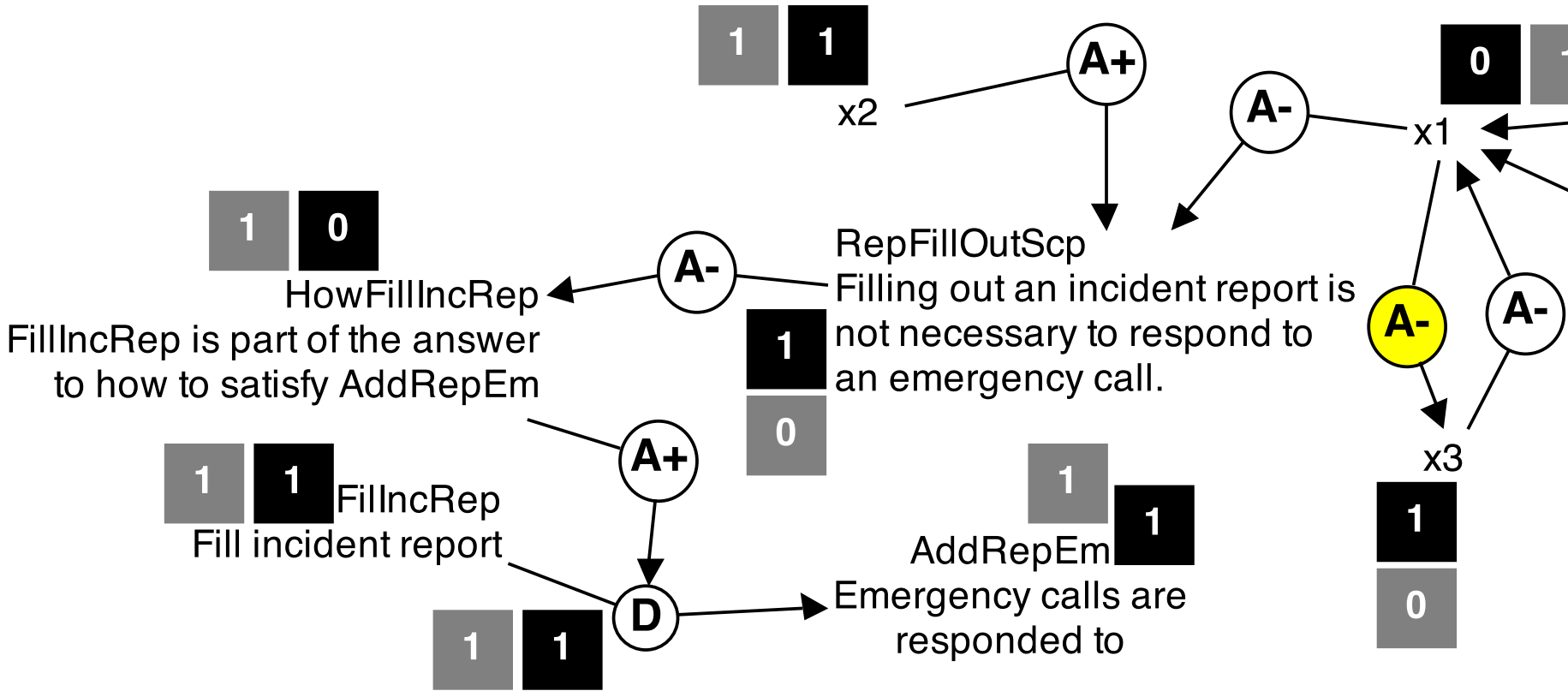
Example: What if there is mutual defeat?



Example: What if there is mutual defeat?



Example: What if there is mutual defeat?



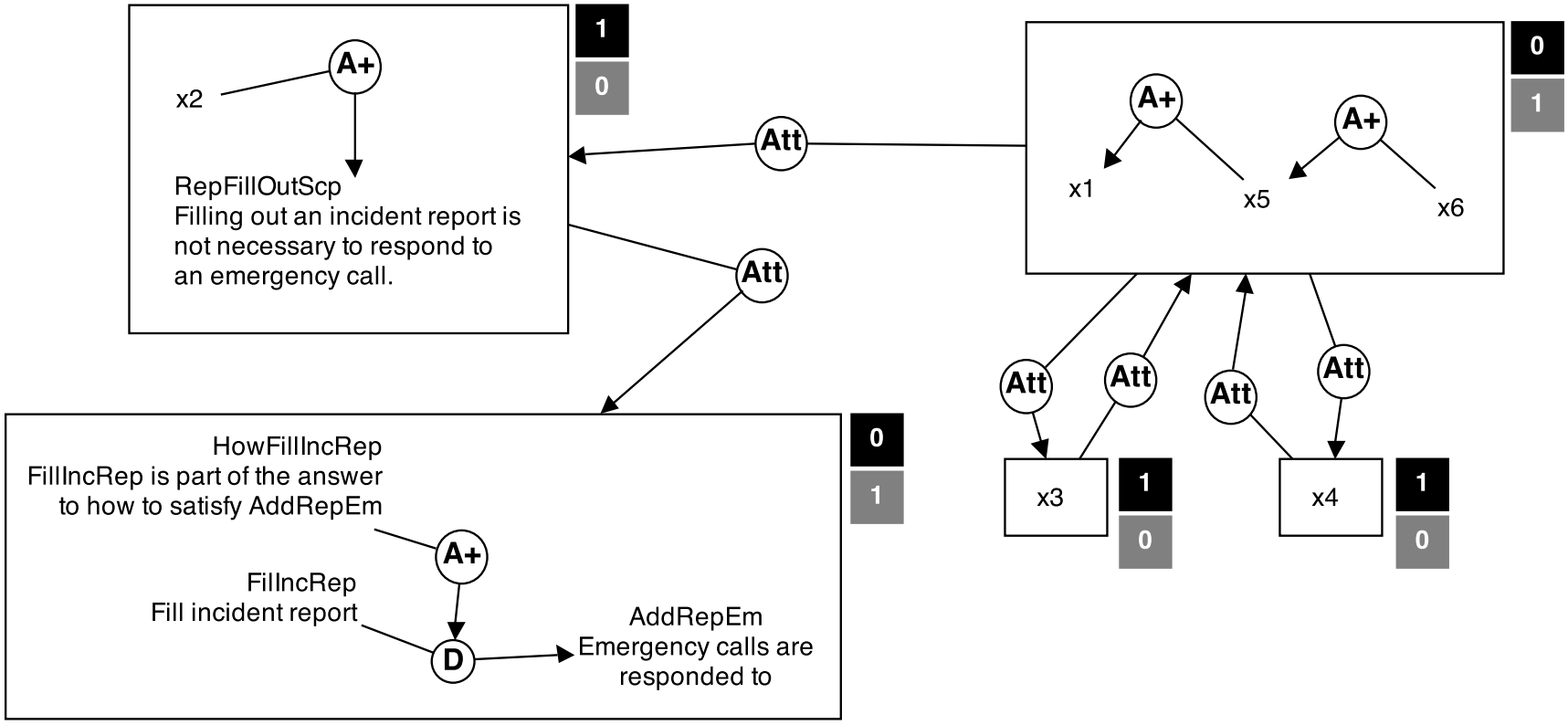
To deliver s.IsAcceptable, map to argumentation frameworks

An argumentation framework has:

- A set of “arguments”,
- A set of “attack” relations over arguments.

And you then look for “extensions”.

Example: An argumentation framework, two extensions shown



How to combine relations in a language?

When you have two or more relations in a language, you need to decide if, and how to use them **together**.

That is, you need to decide:

- Are instances of **different** relations allowed over **same** Fragments?
- If yes, then
  - Which of them are **modelling errors**?
  - Which of them are **not modelling errors**? What are they for?

Let's make this more concrete

Suppose a language can represent these relations over Fragments:

- Inform (r.ifm),
- Positive influence (r.inf.pos), and
- Negative influence (r.inf.neg).

Here is a definition of that language

Language: Achernar

Achernar

Language Modules

F, r.ifm, r.inf.pos, r.inf.neg, f.map.abrel.g

Domain

Set F of Fragments, r.ifm $\subseteq F \times F$, r.inf.pos $\subseteq F \times F$, and r.inf.neg $\subseteq F \times F$.

Syntax

Same as L.Alpheratz.

Mapping

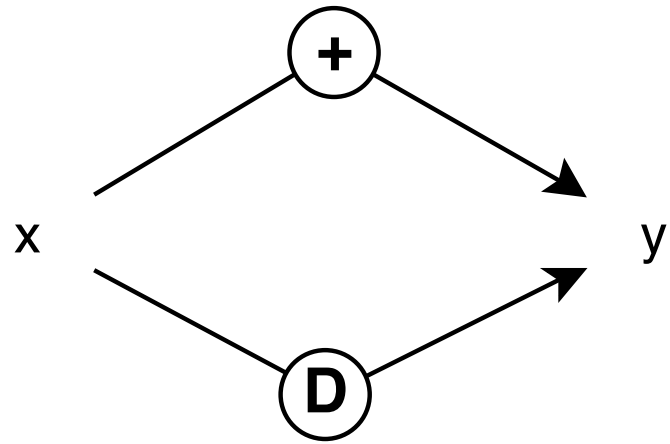
$\mathcal{D}(\alpha) \in F$ and

$$\mathcal{D}(\beta) \in \text{r.ifm} \cup \text{r.inf.pos} \cup \text{r.inf.neg}.$$

Language Services

Same as r.ifm, r.inf.pos, and r.inf.neg.

Consider this model



If relations are independent

...then the language definition ignores relation interactions.

The language user has to figure out what to do with interactions.

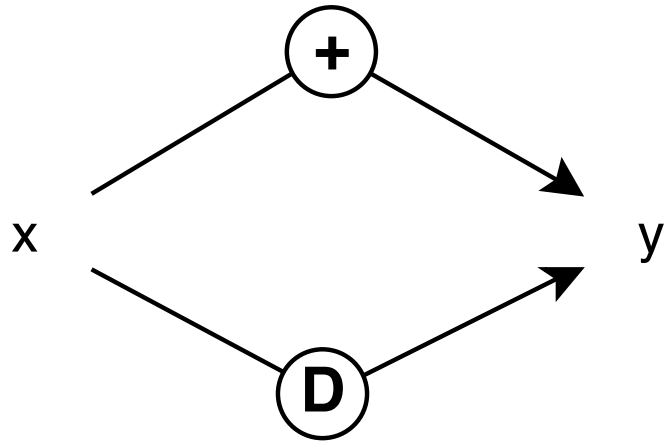
If relations are not independent

Exercise 11 : Define rules for how r.inf.pos, r.inf.neg, and r.ifm interact

Suppose a model can represent positive and negative influence, and inform relations between Fragments. Consider all possible combinations of relation instances between two Fragments, and define rules for what to do in each case.

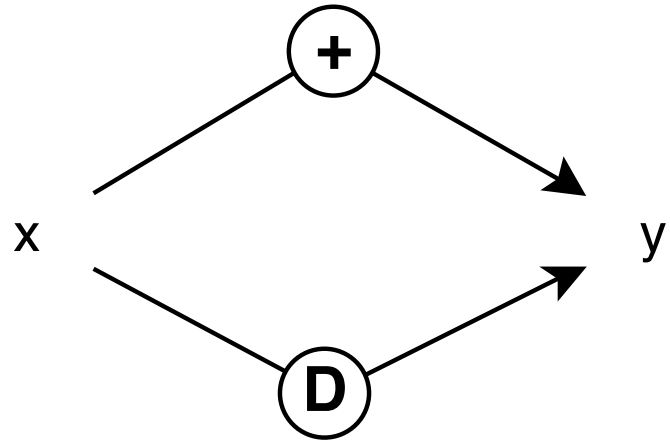
- How would you define these rules in a language?
- How would you define a new language from L.Achernar which includes these rules?

Case 1



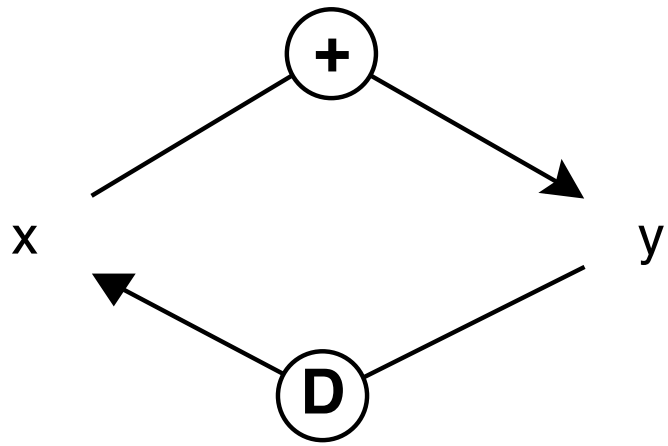
What can you say in this case?

Case 1



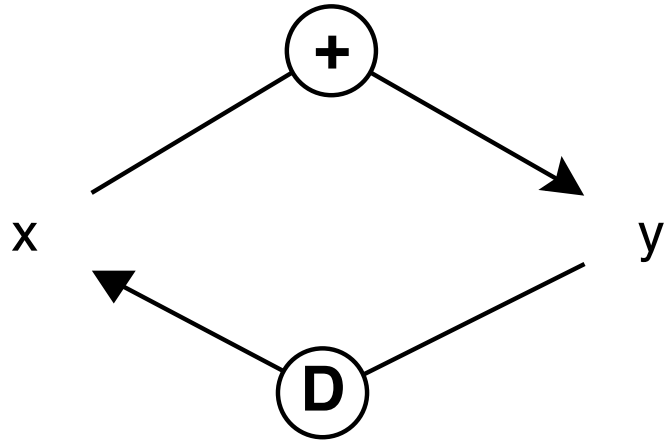
The case is is allowed, and indicates that *x* informs *y*, and in such a way that satisfying it positively influences the satisfaction of *y*.

Case 2



What can you say in this case?

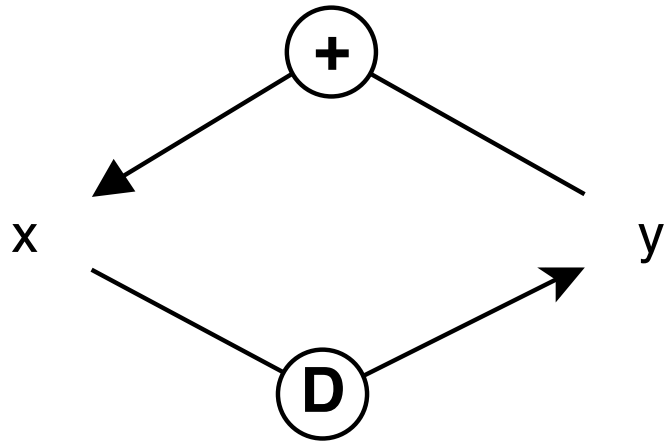
Case 2



Different approaches:

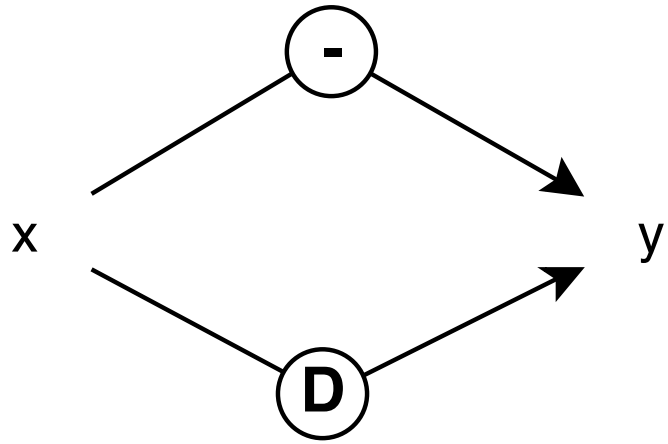
- Not allowed, and one of the two should be removed from the model.
- Allow this if *y* adds such details to *x* by explaining the consequences which will occur if *x* is not satisfied, so that if *x* is satisfied, these consequences will occur, which is captured by the positive influence relation.

Case 3



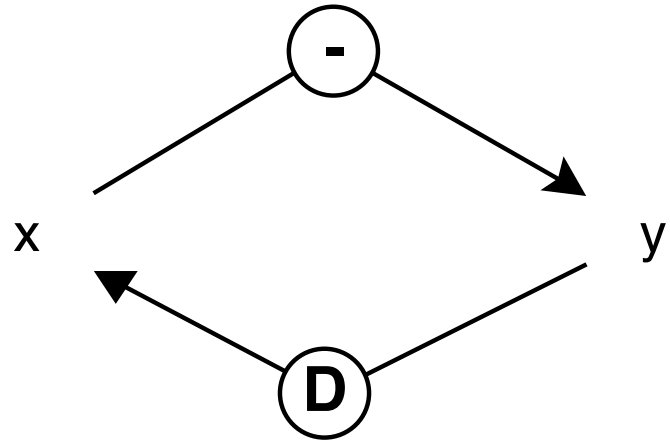
Same as Case 2.

Case 4



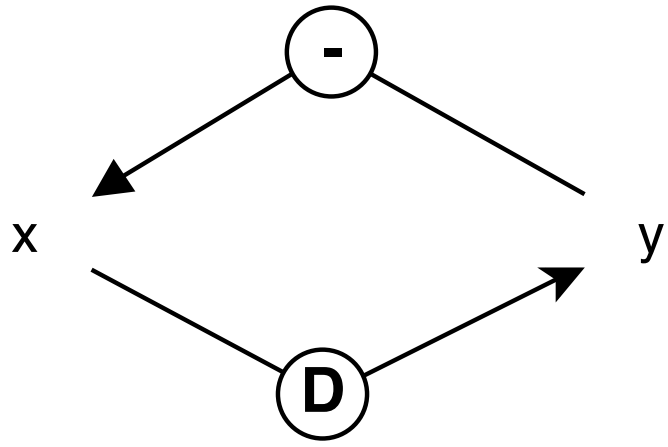
What can you say in this case?

Case 5



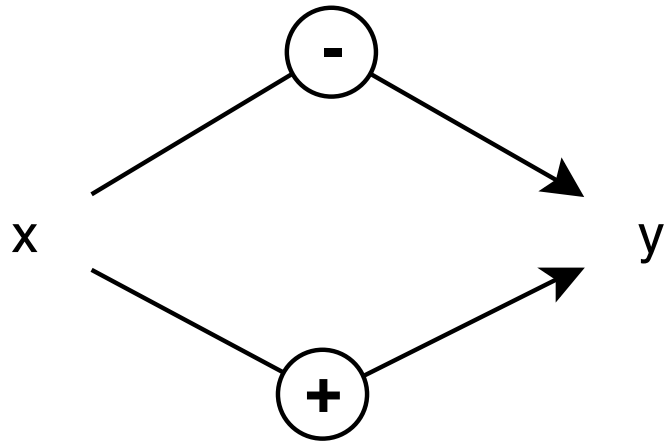
Analogous options to those for $(y, x) \in$ r.ifm and $(x, y) \in$ r.inf.pos, except that there is negative influence.

Case 6



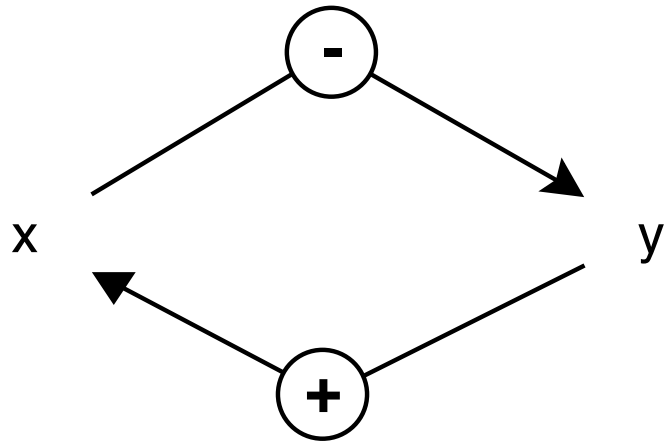
Same as Case 5.

Case 7



Not allowed, and one of the two should be removed.

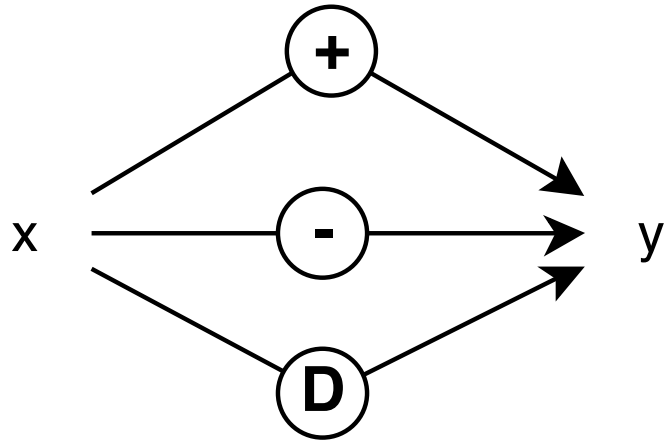
Case 8



Different approaches:

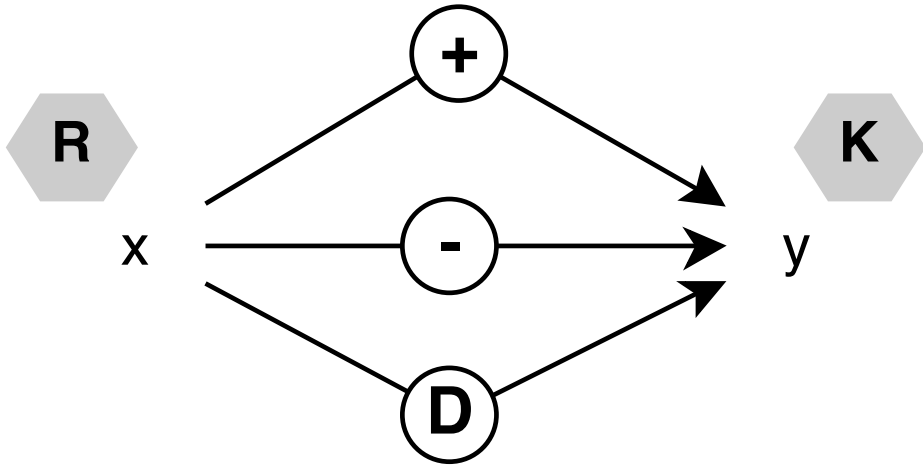
- Remove one of the two influence relations.
- Consider that these two influence relations represent a feedback mechanism, and leave them in the model.

Case 9



What would you say in this case?
Anything new relative to Cases 1 to 8?

Case 10



What would you say in this case?
Anything new relative to Case 9?

Guidelines

Outline

1. How to find guidelines?
2. How to combine guidelines?
3. How to strengthen or weaken guidelines?

Guideline example

“Add details to the model until all stakeholders have agreed that the most detailed elements are detailed enough.”

Consider these Language Services

Language Service

HowToAddDetails: Given a Fragment x, **how to find a new Fragment** y which adds details to x, that is, such that $(y, x) \in$ r.ifm?

Language Service

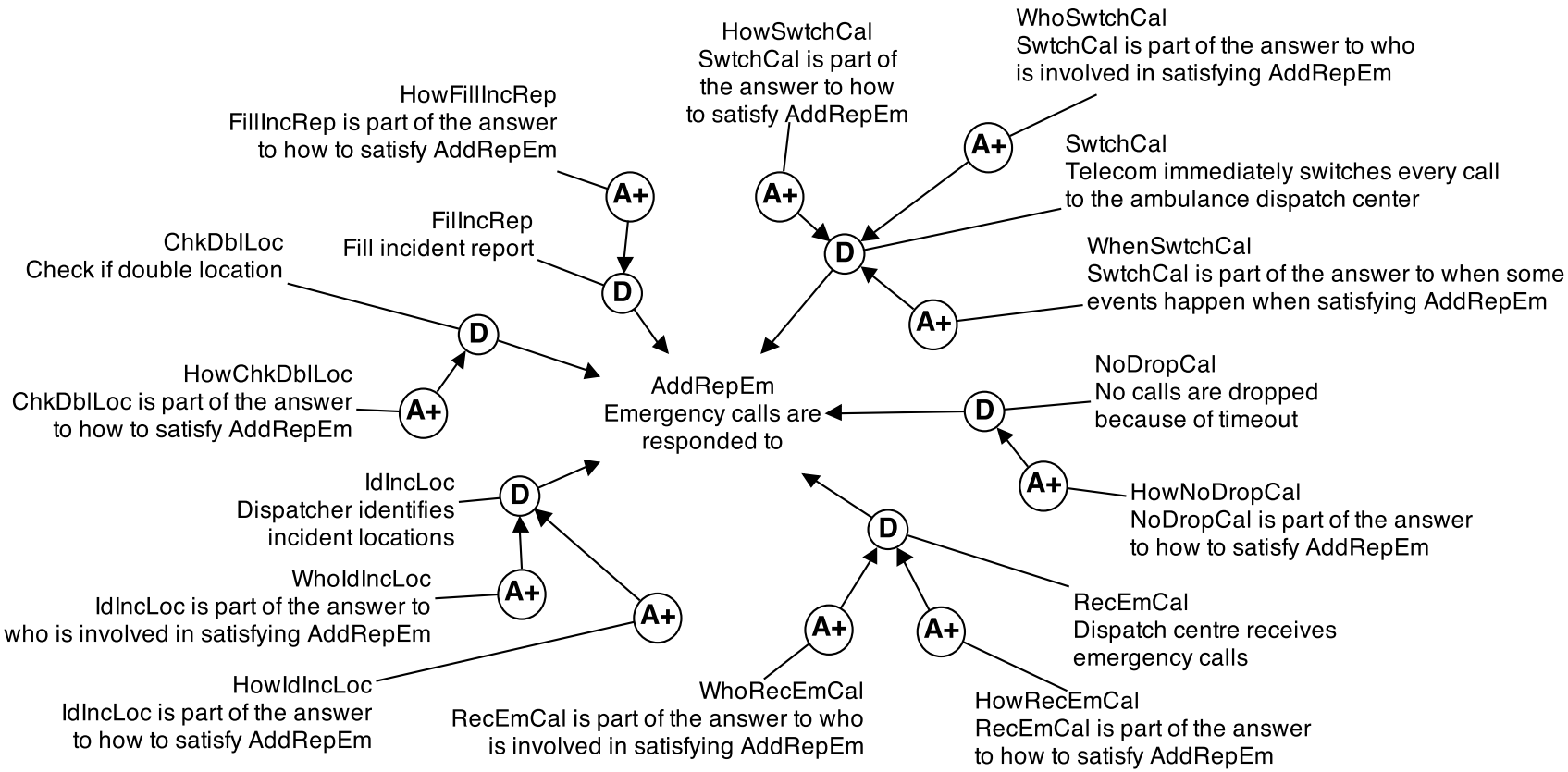
WhyAddsDetails: Given two Fragments x and y such that $(y, x) \in$ r.ifm, **why** does y add details to x?

Exercise 12: Define a language which delivers s.HowToAddDetails and s.WhyAddsDetails

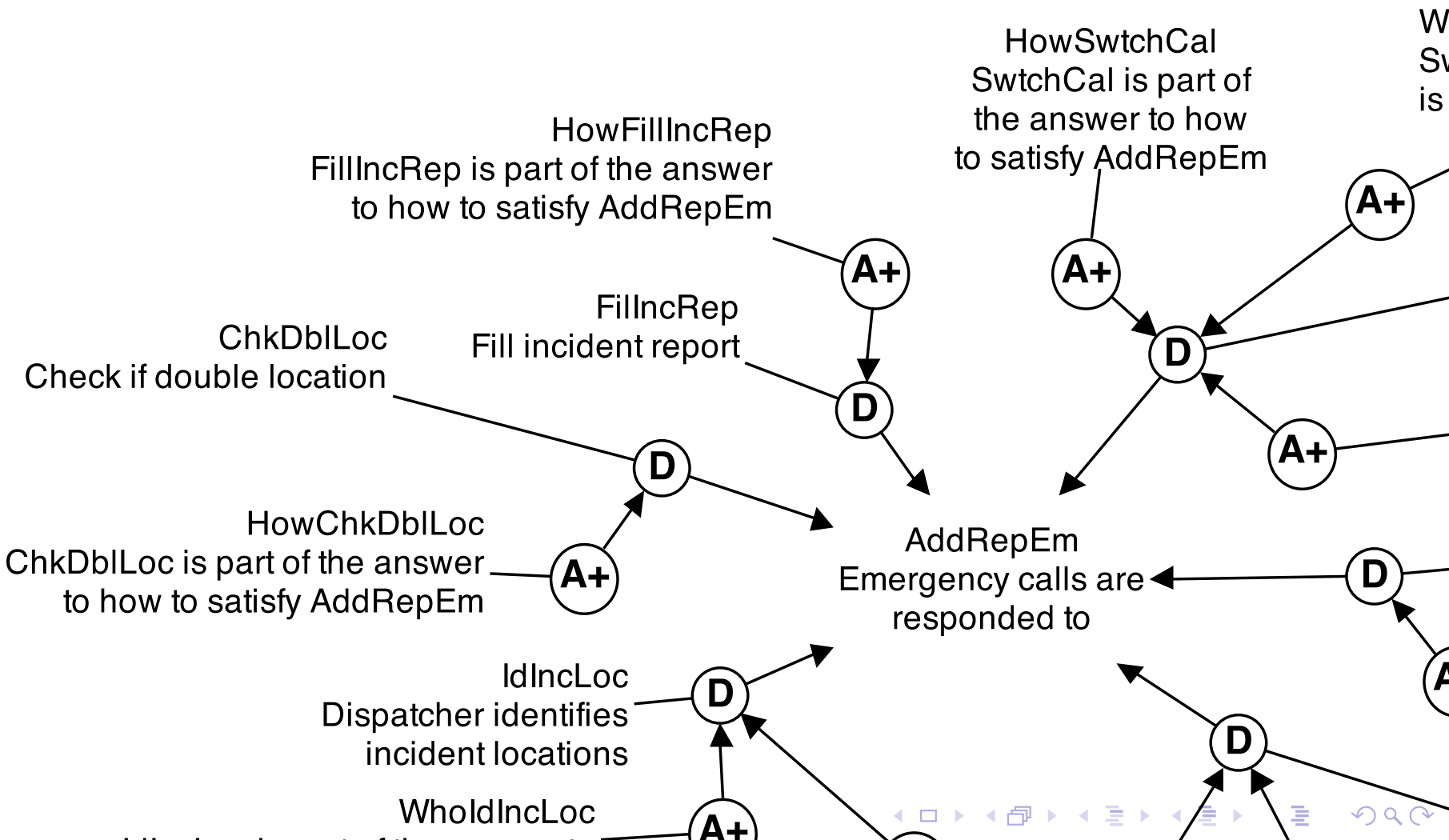
Start with these:

- What does a language need to deliver these?
- Does it need new relations, new functions, or otherwise?
- How are these new relations or functions related to r.ifm?

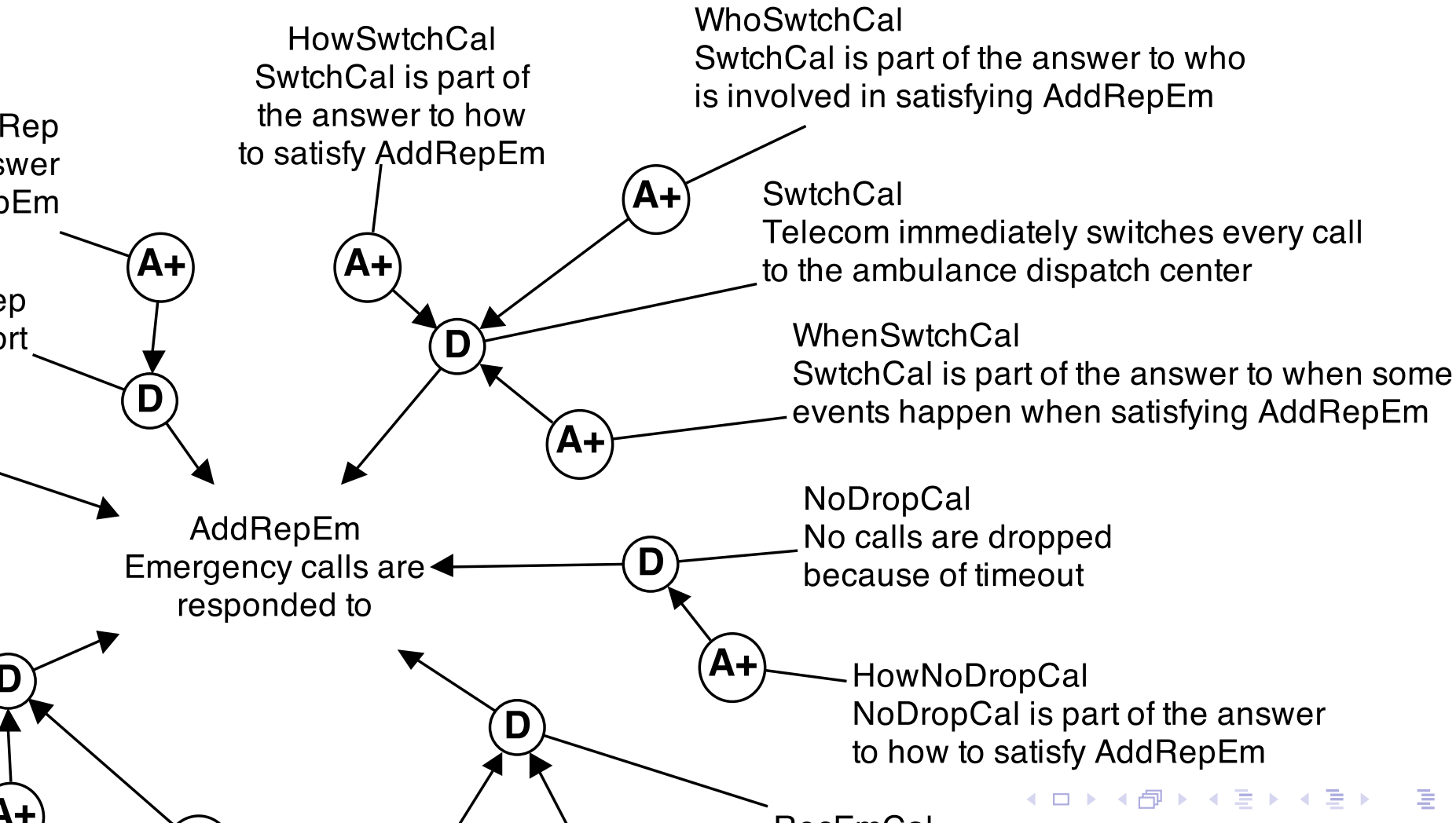
Does the model answer s.WhyAddsDetails?



Does the model answer s.WhyAddsDetails?



Does the model answer s.WhyAddsDetails?



Are there guidelines in recurring arguments?

Recurring arguments suggest questions to ask for any Fragment:

- *Who*: Who does (satisfies) *x*?
- *How*: How is *x* done (satisfied)?
- *When*: When is *x* done (satisfied)?
- *Where*: Where is *x* done (satisfied)?
- *WhoFor*: Who needs *x* to be done (satisfied)?

Questions in a function

Function

Add details
(f.add.ifm)

Input

Fragment x.

Do

1. Ask the following questions about x:
   - *Who*: Who does (satisfies) x?
   - *How*: How is x done (satisfied)?
   - *When*: When is x done (satisfied)?
   - *Where*: Where is x done (satisfied)?
   - *WhoFor*: Who needs x to be done (satisfied)?
2. Define sets F_q and R_q, for $q \in \{Who, How, When, Where, WhoFor\}$, such that:
   - 2.1 each $y \in \mathrm{F}_q$ answers the question q for x,
   - 2.2 if y answers the question q for x, then there is $(y, x) \in$ r.ifm in R_q.

Questions in a function

Function

Add details
(f.add.ifm)

Input

Fragment x.

Do

...

Output

Sets F_q and R_q, for $q \in \{Who, How, When, Where, WhoFor\}$.

Language Services

- s.HowToAddDetails: Apply f.add.ifm to Fragments in a given model M, and add the resulting sets back to M.
- s.WhyAddsDetails: If R_q is output by applying f.AddsDetails, and $(y, x) \in R_q$, then y adds details to x because it answers the question q for x.

Why not use only f.add.fm?

The function adds new Fragments.

How to represent in models, that these new Fragments answer questions about existing Fragments?

Keeping answers in models

Relation

Answers question q (r.q)

Domain & Dimension

r.q $\subseteq R$, where R is a set of r.ifm instances.

Properties

None.

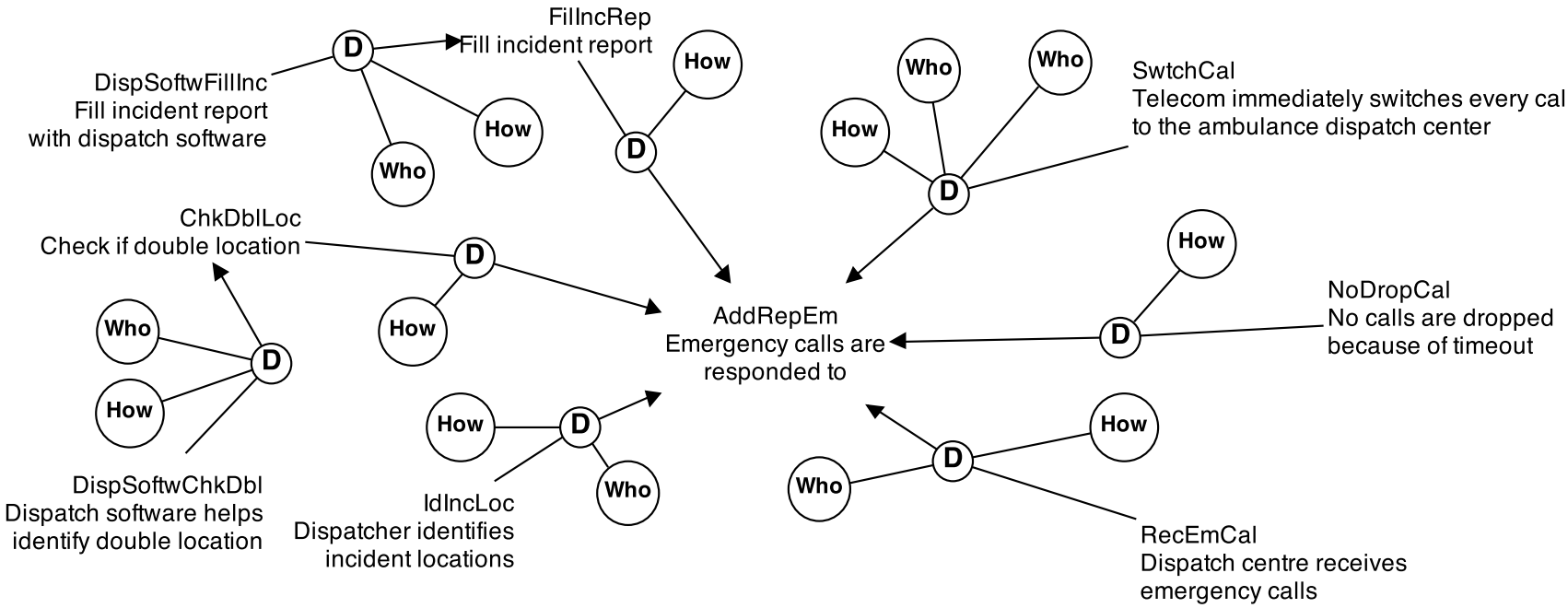
Reading

$r \in$ r.q, where $r = (y, x)$, reads "y adds details to x by answering question q for x".

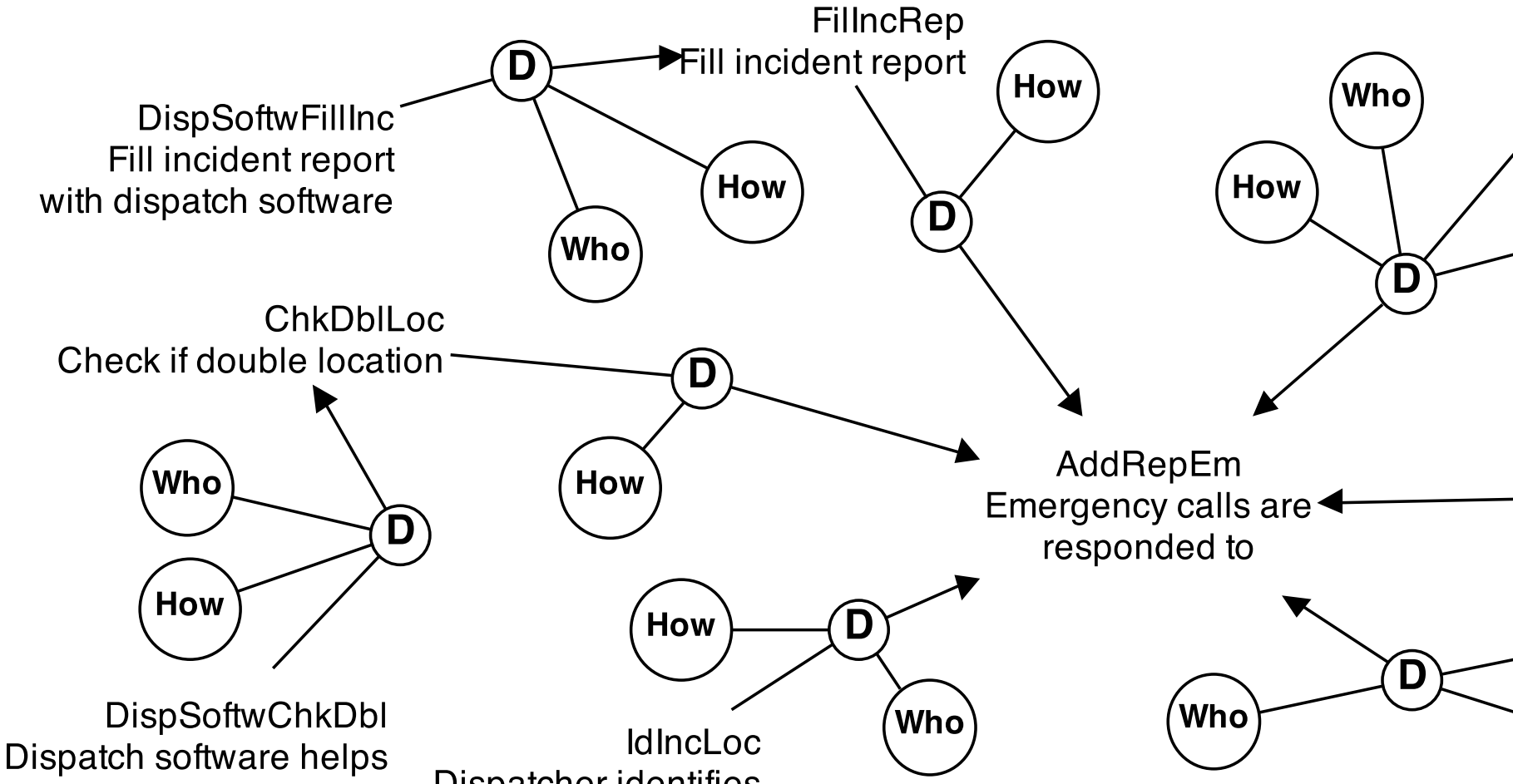
Language Services

- s.WhyAddsDetails: If $(y, x) \in$ r.q, then y adds details to x because it answers the question q for x.

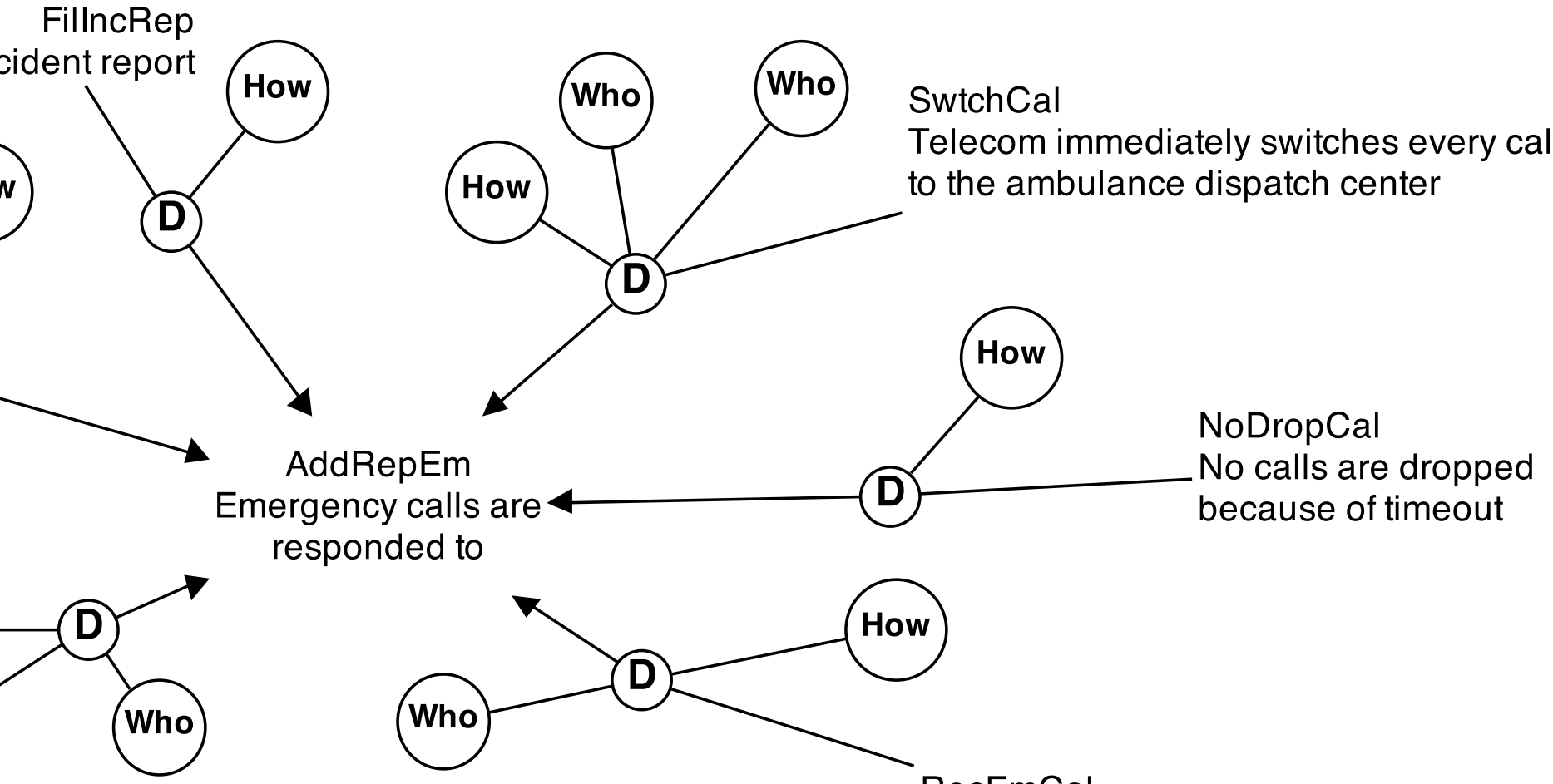
Example: A model



Example: Zoom in



Example: Zoom in



What is the language of this model?

Language
Hamal
Language Modules
F, r.ifm, r.Who, r.How, r.When, r.Where, r.WhoFor, f.add.ifm

What is the language of this model?

Language

Hamal

Syntax

A model M in the language is a set of symbols $M = \{\phi_1, \ldots, \phi_n\}$, where every ϕ is generated according to the following BNF rules:

$$
\begin{array}{lcl}
\alpha & ::= & x \mid y \mid z \mid \ldots \\
\beta & ::= & (\alpha, \alpha) \\
\gamma & ::= & \mathit{Who} \mid \mathit{How} \mid \mathit{When} \mid \mathit{Where} \mid \mathit{WhoFor} \\
\delta & ::= & \gamma.\beta \\
\phi & ::= & \alpha \mid \beta \mid \delta
\end{array}
$$

What is the language of this model?

Language

Hamal

Domain

F is a set of Fragments. r.ifm $\subseteq F \times F$, r.q $\in$ r.ifm, for every

$$q \in \{Who, How, When, Where, WhoFor\}$$

What is the language of this model?

Language

Hamal

Mapping

α symbols denote Fragments, $\mathcal{D}(\alpha) \in F$. β are for r.ifm instances, that is, $\mathcal{D}(\beta) \in$ r.ifm. δ symbols denote r.q instances, $\mathcal{D}((Who.\beta)) \in$ r.Who, $\ldots$, $\mathcal{D}((WhoFor.\beta)) \in$ r.WhoFor.

What is the language of this model?

Language

Hamal

Language Services

- s.WhyAddsDetails: If $q.(y, x) \in$ r.q, then y adds details to x because it answers the question q for x.

How to make composite guidelines?

A composite guideline, when used, involves the application of other guidelines.

Exercise 13: Define a function for operationalisation

Define a function which checks if a Fragment in a model is operationalised, according to the rule *Op* in the text.

- Which relations do you need to have over Fragments in a model, in order to check if a Fragment is operationalised?
- Are there other guidelines which this operationalisation function uses?

You want to deliver this

Language Service

AreOpr: Are all Fragments in *W* operationalised?

Add a function

Function

Operationalise all Fragments in a set
(f.opr.all)

Input

Set W of Fragments.

Do

1. Let X be an empty set, add all members of W to X.
2. Apply f.add.ifm to every Fragment $w \in X$, and add to X all new Fragments which you thereby find. **When to stop doing this?**
3. **?**
4. **?**

Output

Set Z of Fragments which are said to operationalise all Fragments in W.

Language Services

- s.AreOpr: Yes, if there is a set Z made by applying f.opr.all to W.

Add a function

Function

Operationalise all Fragments in a set
(f.opr.all)

Input

Set W of Fragments.

Do

1. Let X be an empty set, add all members of W to X.
2. Apply f.add.ifm to every Fragment $w \in X$, and add to X all new Fragments which you thereby find. **If a Fragment $y \in X$ is detailed enough that it is known how to satisfy and, or execute what it describes, and it is known who takes the responsibility to do so, then do not apply f.add.ifm to y.**
3. **?**
4. **?**

Output

Set Z of Fragments which are said to operationalise all Fragments in W.

Add a function

Function

Operationalise all Fragments in a set
(f.opr.all)

Input

Set W of Fragments.

Do

1. ...
2. ...
3. For every $(a, b) \in$ r.ifm, where $a, b \in X$, check if there should be $(a, b) \in$ r.inf.pos or $(a, b) \in$ r.inf.neg and if yes, then add it. Stop when it is known how the satisfaction of each more detailed Fragment influences the satisfaction of the Fragment to which adds details.
4. **?**

Output

Set Z of Fragments which are said to operationalise all Fragments in W.

Language Services

Add a function

Function

Operationalise all Fragments in a set
(f.opr.all)

Input

Set W of Fragments.

Do

1. ...
2. ...
3. ...
4. If there is a set $Z \subseteq X$ such that satisfying all Fragments in Z positively influences the satisfaction of all Fragments in W, and there are no Fragments in $W \setminus Z$ which inform those in Z, then stop. Otherwise, go back to step 1 above.

Output

Set Z of Fragments which are said to operationalise all Fragments in W.

Language Services

Categories

What do I mean by “category”?

A category groups Fragments which share the same properties, and thus distinguishes these same Fragments from others which do not.

Outline for categories

1. Why and how to use independent categories?
2. What to do when there is a taxonomy of categories?
3. What is the meta-model, and what the ontology of a language?
4. Why and how to define derived categories and relations?
5. How to enforce the intended use of categories?

Recall the Default RP

Default RP is:

Given:

- a set R of requirements, and
- a set K of domain knowledge,

Define a set S called "specification", which satisfies

- the provability condition $K, S \vdash R$, and
- the consistency condition $K, S \nvdash \bot$.

Exercise 14 : Define the minimal set of categories needed to represent those of the Default RP

Define the minimal set of categories which a language would need, to make the distinctions in the Default RP.

Questions to consider

- How many categories are needed?
- What are the properties which decide if a Fragment is in one of these categories?
- Can a Fragment be in two or more of these categories?
- If yes, which conditions does it have to satisfy? If not, then why not?

Language Module for categories

What slots should be in the Language Module for a category?

Language Module should answer at least:

- What is being categorised?
- What to satisfy, to be in a category?
- Category name / reading?
- Language Services?

Requirements category

Category

Requirement (c.r)

Domain

c.Requirement $\subseteq$ F, where F is a set of Fragments.

Membership conditions

?

Reading

?

Language Services

?

What did Zave and Jackson say about requirements?

> *"The primary distinction necessary for requirements engineering is captured by two grammatical moods. Statements in the 'indicative' mood describe the environment as it is in the absence of the machine or regardless of the actions of the machine; these statements are often called 'assumptions' or 'domain knowledge.'* ***Statements in the 'optative' mood describe the environment as we would like it to be and as we hope it will be when the machine is connected to the environment. Optative statements are commonly called 'requirements.'*** *[...] A specification is also an optative property, but one that must be implementable."*

Zave and Jackson, "Four dark corners of requirements engineering", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(1):1-30, 1997.

Requirements category

Category

Requirement (c.r)

Domain

c.Requirement $\subseteq$ F, where F is a set of Fragments.

Membership conditions

x is in the optative mood, and describes “the environment as we would like it to be and as we hope it will be when the machine is connected to the environment” [**?**].

Reading

$x \in$ c.r reads “*x* is a requirement”.

Language Services

- **s.IsReq**: Is *x* a requirement? Yes, if $x \in$ c.r.

Domain knowledge category

Category

Domain knowledge (c.k)

Domain

c.Domain knowledge $\subseteq$ F, where F is a set of Fragments.

Membership conditions

x is in indicative mood and describes “the environment as it is in the absence of the machine or regardless of the actions of the machine” [**?**].

Reading

$x \in$ c.k reads “*x* is domain knowledge”.

Language Services

- **s.IsDomK**: Is *x* domain knowledge? Yes, if $x \in$ c.k.

Specification category

Category

Specification (c.s)

Domain

c.Specification $\subseteq$ F, where F is a set of Fragments.

Membership conditions

x is a statement in optative, which is implementable, that is, it is known who and how will do what the statement says.

Reading

$x \in$ c.s reads "x is a specification".

Language Services

- **s.IsSpec**: Is x a specification? Yes, if $x \in$ c.s.

How to use the three categories together?

Try delivering the following:

Language Service

WhichKSR: Which Fragments in X are requirements, which are domain knowledge, and which are specifications?

You can make a function

Function

Categorise in Default RP categories
(f.cat.ksr)

Input

Set X of Fragments.

Do

For each $x \in X$:

- if x is in c.r, then let $cat(x) =$ c.r, else
- if x is in c.k, then let $cat(x) =$ c.k, else
- if x is in c.s, then let $cat(x) =$ c.s.

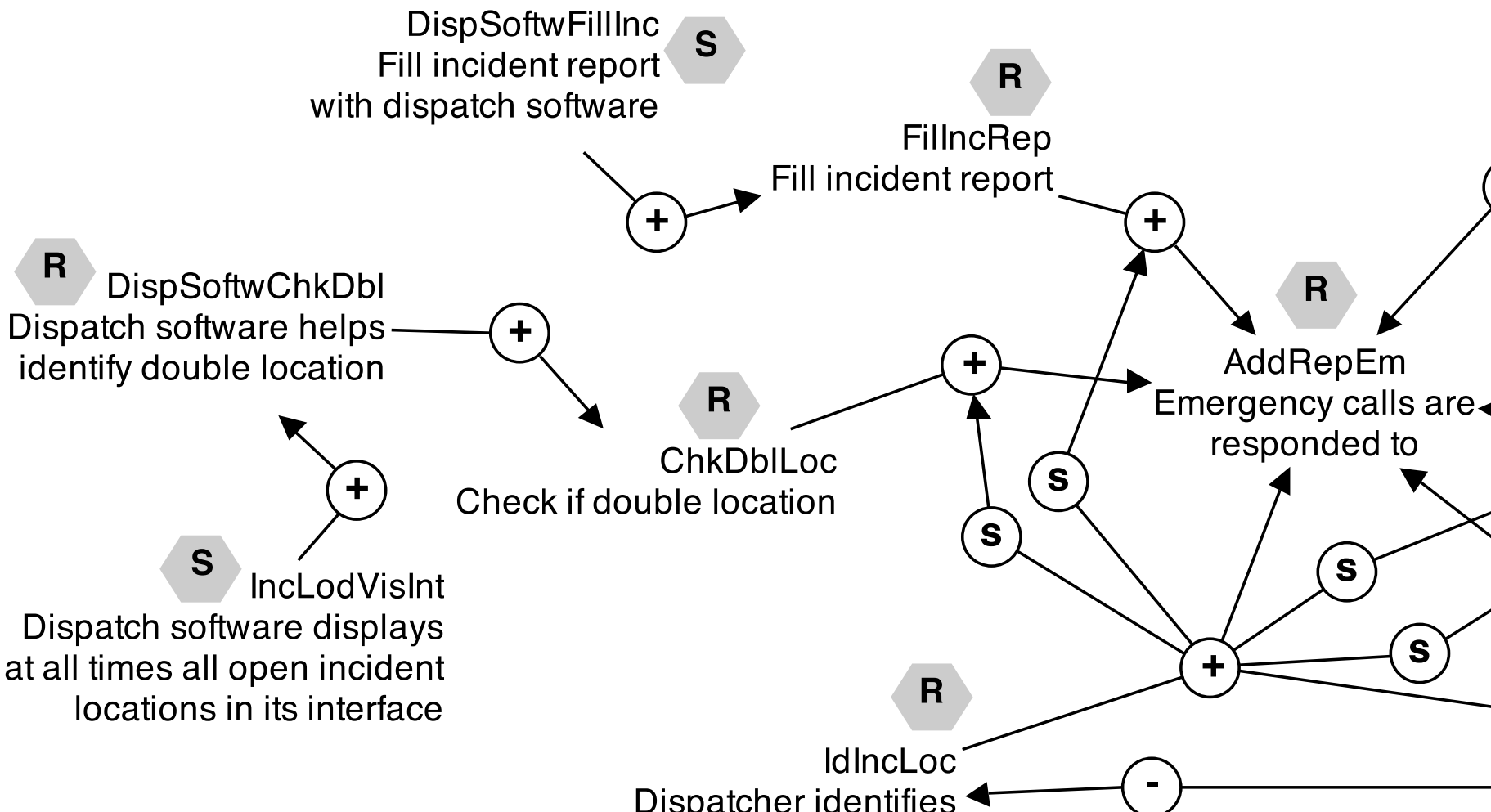
Output

Function *ksr*.

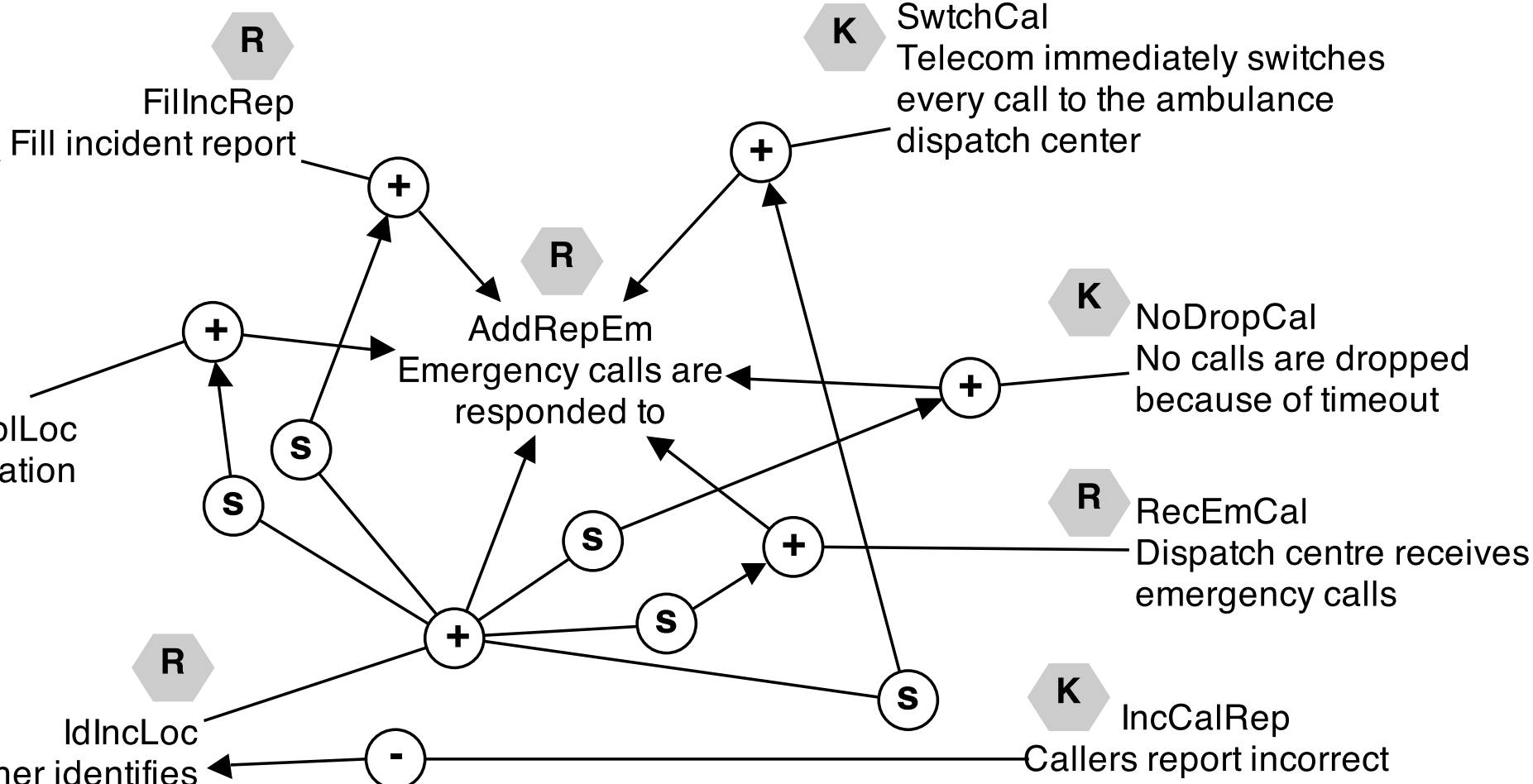
Language Services

- s.WhichKSR: Function *ksr* says, for each Fragment in X, if it is a requirement, domain knowledge, or specification.

A model in a language which has f.cat.ksr



A model in a language which has f.cat.ksr



Which language is this model in?

- What relations are there?
- Which categories?
- What is the syntax?
- Domain?
- Mapping?

Language Modules in the language

Language: Menkar

Menkar

Language Modules

F, r.inf.pos, r.inf.neg, r.str.inf, f.map.abrel.g, f.cat.ksr

Domain

?

Syntax

?

Mapping

?

Language Services

?

Syntax

Language

Menkar

Syntax

A model M in the language is a set of symbols $M = \{\phi_1, \ldots, \phi_n\}$, where every ϕ is generated according to the following BNF rules:

$$
\begin{aligned}
\alpha &::= x \mid y \mid z \mid \ldots \\
\beta &::= r \mid k \mid s \\
\gamma &::= \beta(\alpha) \\
\delta &::= (\gamma, \gamma) \\
\epsilon &::= (\delta, \delta) \\
\phi &::= \gamma \mid \delta \mid \epsilon
\end{aligned}
$$

Domain

Language

Menkar

Domain

Fragments are partitioned onto requirements, domain knowledge, and specifications, that is, $F = c.r \cup c.k \cup c.s$ and $c.r \cap c.k \cap c.s = \emptyset$. Influence relations are over Fragments, $r.inf.pos \subseteq F \times F$, $r.inf.neg \subseteq F \times F$. Relative strength of influence is a relation over influence relations of the same type:

$$r.str.inf \subseteq (r.inf.pos \times r.inf.pos) \cup (r.inf.neg \times r.inf.neg).$$

Mapping

Language

Menkar

Mapping

α symbols denote uncategorised Fragments. γ symbols denote categorised Fragments, so $\mathcal{D}(r(\alpha)) \in$ c.r, $\mathcal{D}(k(\alpha)) \in$ c.k, and $\mathcal{D}(s(\alpha)) \in$ c.s. δ symbols denote influence relations, $\mathcal{D}(\delta) \in$ r.inf.pos $\cup$ r.inf.neg. ϵ symbols denote relative strength of influence, $\mathcal{D}(\epsilon) \in$ r.str.inf.

Language Services

Language
Menkar
Language Services Those of r.inf.pos, r.inf.neg, r.str.inf, f.map.abrel.g, and f.cat.ksr.

How to define a taxonomy of categories?

- You restrict the domain of the child category.
- You avoid clashes with properties of the parent category.

Exercise 15 : Define categories which are specialisations of an existing category in a language

Choose an existing category in L.Menkar and identify at least two categories which are its specialisations (its sub-categories).

Functional and nonfunctional requirements

Suppose that:

- A requirement is **functional** if it can either be satisfied or not.
- A requirement is **nonfunctional** if:
  - it can be partially satisfied, and
  - different stakeholders can have different judgments of its satisfaction level.

Functional requirement

Category

Functional requirement (c.r.f)

Domain

c.Functional requirement $\subseteq X$, where $X \subseteq$ **c.r** .

Membership conditions

x is a member of c.r such that it is either satisfied or not.

Reading

$x \in$ c.r.f reads "x is a functional requirement".

Language Services

- **s.IsFunctReq**: Is x a functional requirement? Yes, if $x \in$ c.r.f.

Nonfunctional requirement

Category

Nonfunctional requirement (c.r.nf)

Domain

c.Nonfunctional requirement $\subseteq X$, where $X \subseteq$ **c.r** .

Membership conditions

x is a member of c.r such that it is can be satisfied to some extent, rather than either satisfied or failed, and different stakeholders may judge it to be satisfied to different extents by the same system.

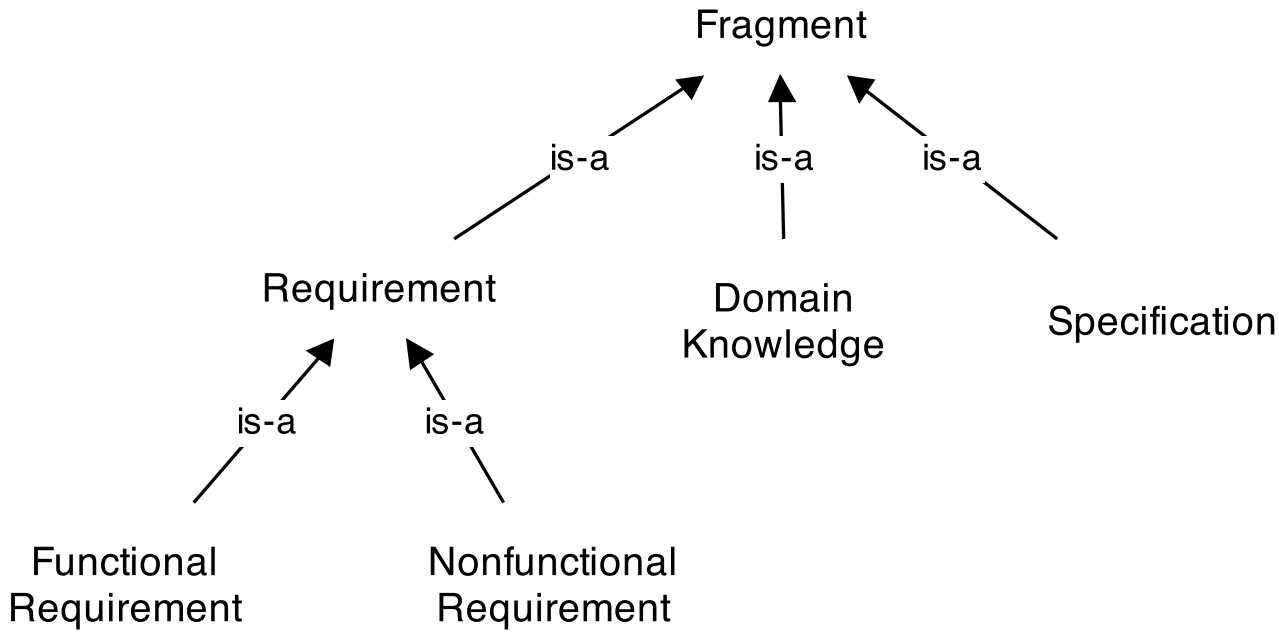
Reading

$x \in$ c.r.nf reads "x is a nonfunctional requirement".

Language Services

- **s.IsNFunctReq**: Is x a nonfunctional requirement? Yes, if $x \in$ c.r.nf.

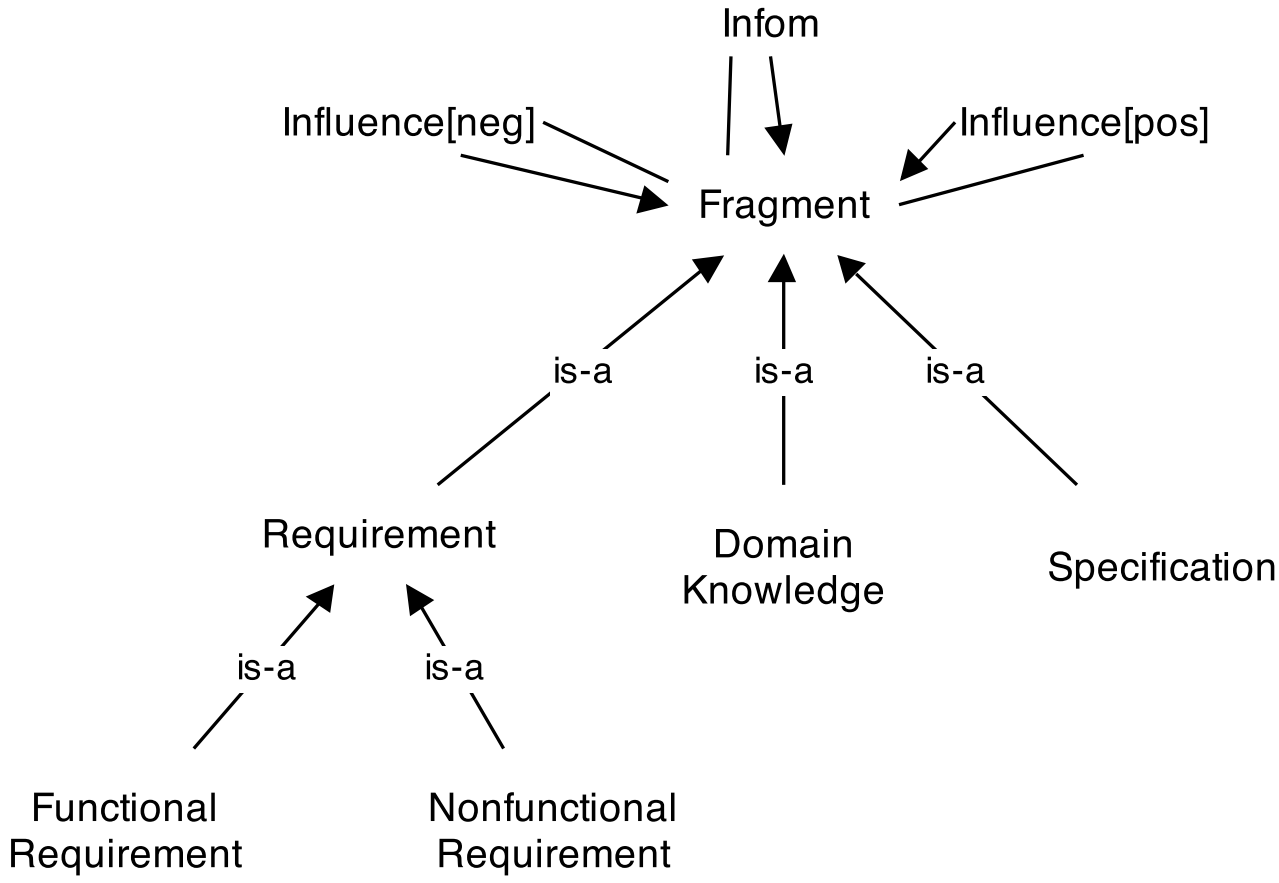
Visualisation of the taxonomy



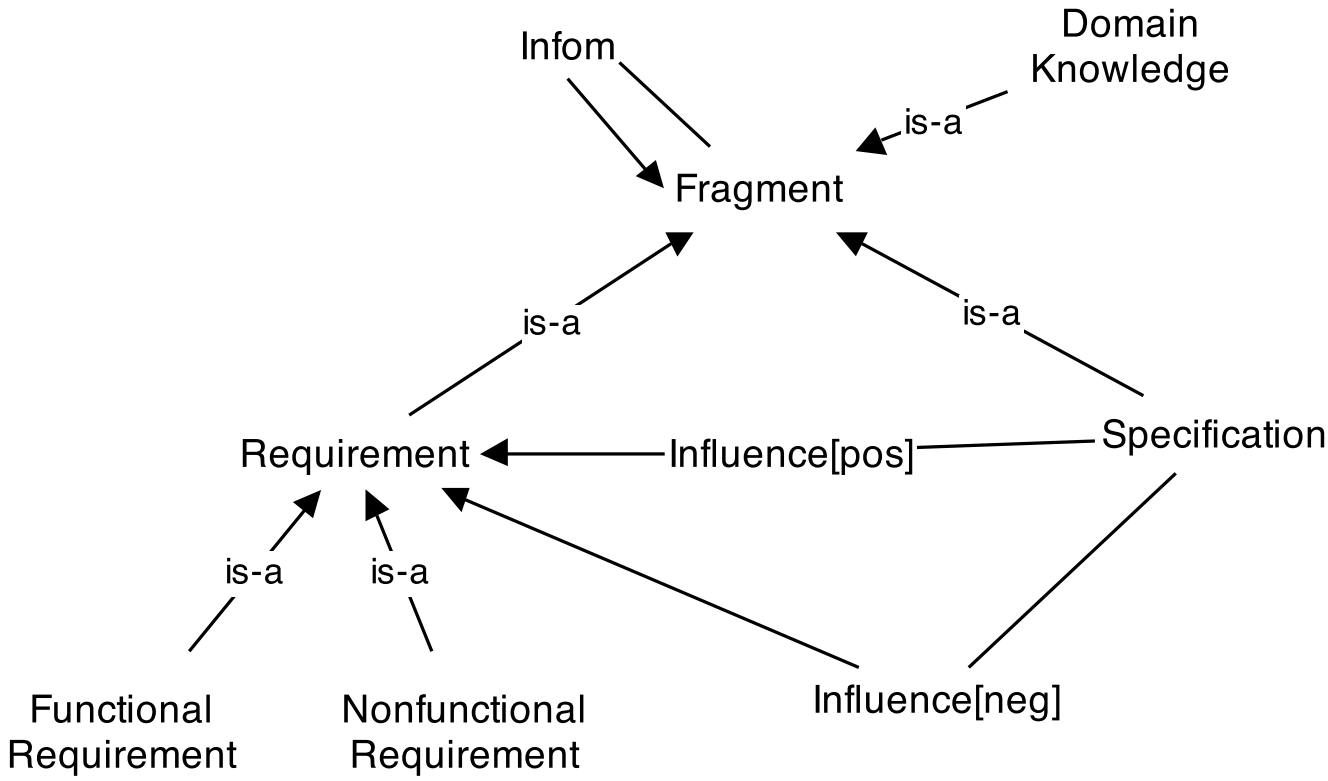
Meta-models and ontologies

- What is the role for an ontology in a language?
- What is the role for a meta-model?
- How are the two different?

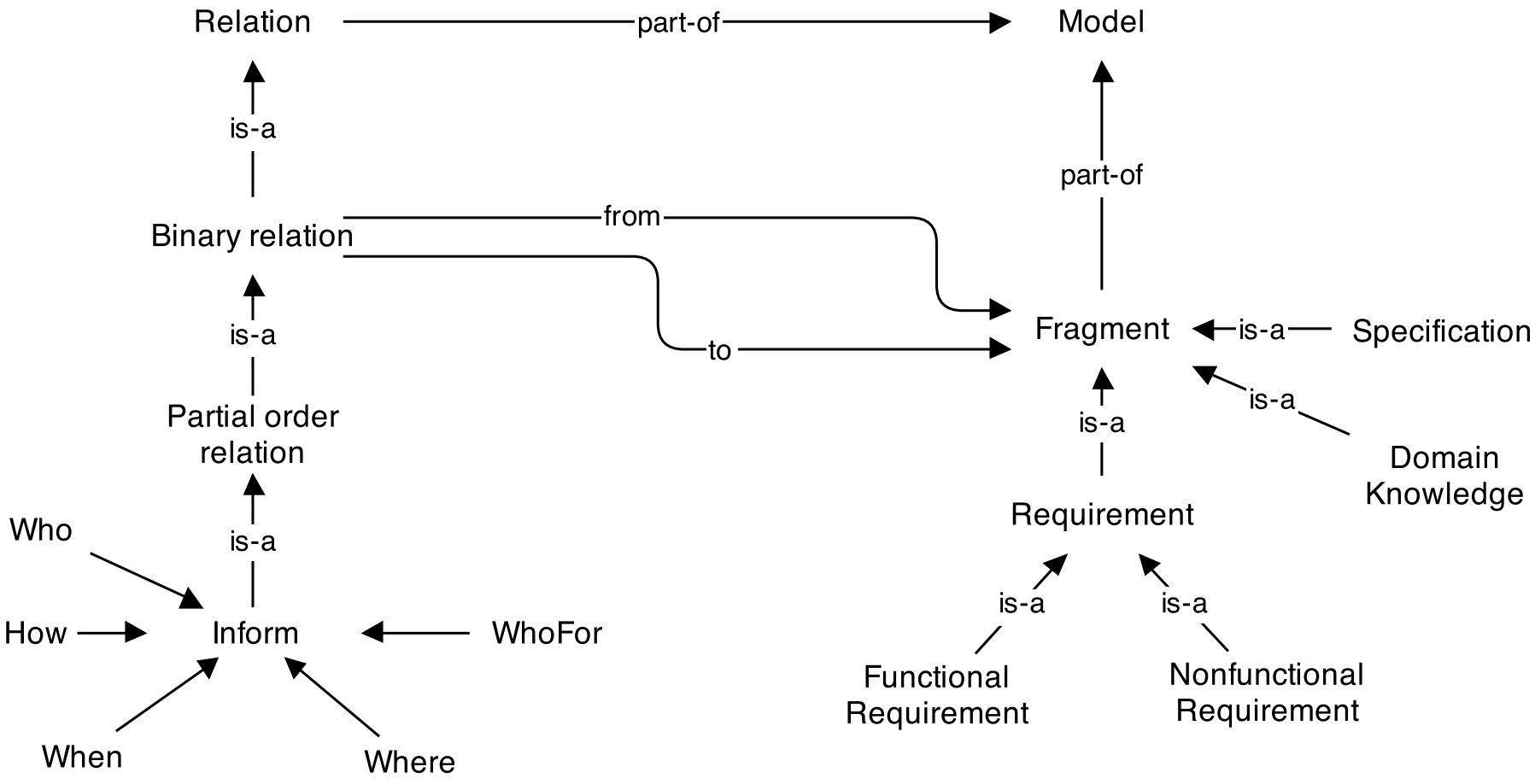
Ontology or meta-model?



Ontology or meta-model?



Ontology or meta-model?



What is a derived category or relation?

When it is defined from existing categories and relations in a language.

Why have derived categories or relations in a language?

How would you define an irrelevant requirement?

Suppose that your model can include irrelevant requirements.

- When is a requirement irrelevant?
- A new category for irrelevant requirements?

Irrelevant requirement

Category

Irrelevant requirement (c.r.irrl)

Domain

c.Irrelevant requirement $\subseteq X$, where $X \subseteq$ c.r.

Membership conditions

x is not acceptable in a given model M according to f.acc.

Reading

$x \in$ c.r.irrl reads "x is an irrelevant requirement".

Language Services

- **s.IsIrrlReq**: Is x an irrelevant requirement? Yes, if $x \in$ c.r.irrl.

How to enforce intended use of categories?

If you know that some Fragments are requirements, then

are all of them operationalised in a model?

If not, you could conclude that the model is incomplete.

A function to check completeness

Function

Completeness of requirements operationalisation (f.chk.rop)

Input

A set X of Fragments, G(X, r.ifm), and G(X, r.inf.pos).

Do

1. Let H be a hypergraph made by merging G(X, r.ifm) and G(X, r.inf.pos).
2. If there is $x \in X$ such that x is in c.r and there is no path in H from $z \in X$ to x, such that z is in c.s, then the model which includes exactly the Fragments in X is incomplete with regards to requirements operationalisation and $v = 1$.

Output

v.

Language Services

- **s.IsROpComp**: Is the model that includes exactly the Fragments X incomplete with regards to requirements operationalisation? : Yes, if $v = 1$, no otherwise

Alternatives and Combinations

Outline

1. Simple Alternatives?
2. Composite Alternatives?
3. Combinations?

Exercise 16: Represent that two relation instances are mutually exclusive

Choose any language defined so far in this tutorial. Without changing that language, how would you represent that two relation instances are mutually exclusive in a model in that language?

Simple Alternative

Mutually exclusive individual Fragments or relation instances. Consider this Language Service:

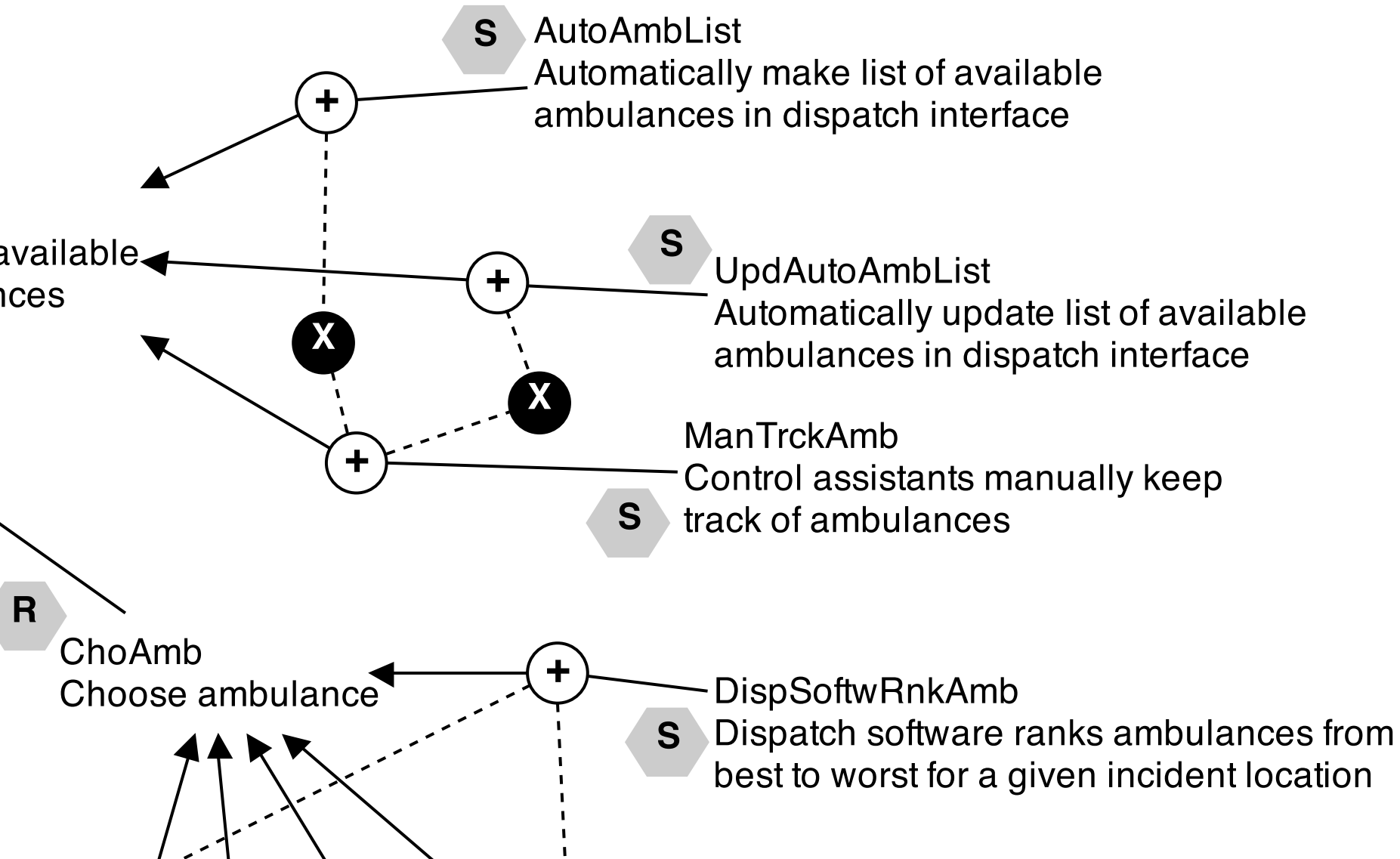
Language Service

InfPosAlt: Which Fragments in the model M show different ways to satisfy x?

Exercise 17: Make a new language from L.Ankaa which can show mutually exclusive relation instances in models

- How would you represent in models that two or more relation instances are mutually exclusive?
- Would you do it with a relation, or otherwise?
- What would you add or remove from L.Ankaa to enable it to show in models that some influence relation instances are mutually exclusive?
- Would the resulting language deliver s.InfPosAlt? If yes, then how?

Example: Mutually exclusive influence relations



XOR relation

Relation

Mutually exclusive relation instances (r.xor)

Domain & Dimension

r.xor $\subseteq R^n$, where R is a set of relation instances.

Properties

If $w \in$ r.xor, then there can be no $e \in$ r.xor such that $e = (\ldots, w, \ldots)$, that is, there can be no r.xor instances over other r.xor instances.

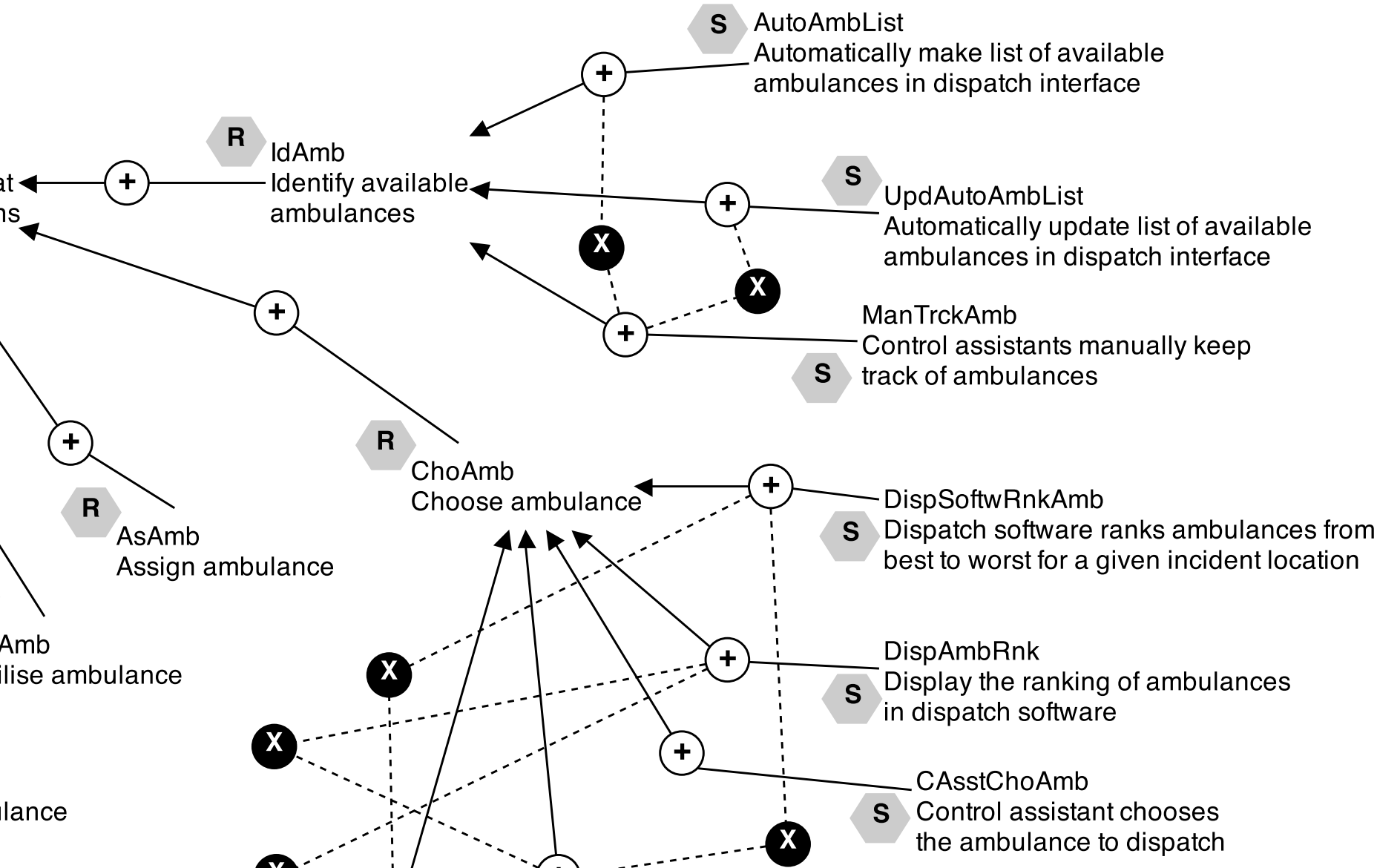
Reading

$(r_1, \ldots, r_n)$r.xor reads “relation instances $r_1, \ldots, r_n$ are mutually exclusive”.

Language Services

- **s.IsAlt**: Are $r_i, \ldots, r_m$ Alternatives? : Yes, if there is $w \in$ r.xor and every $r_i, \ldots, r_m$ is in w.

Define the language of this model



Language with influence and XOR relations

Language: Mirfak

Mirfak

Language Modules

F, r.inf.pos, r.inf.neg, r.xor, f.cat.ksr, f.map.abrel.g

Domain

?

Syntax

?

Mapping

?

Language Services

?

Language with influence and XOR relations

Language

Mirfak

Syntax

A model M in the language is a set of symbols $M = \{\phi_1, \ldots, \phi_n\}$, where every ϕ is generated according to the following BNF rules:

$$
\begin{aligned}
\alpha &::= x \mid y \mid z \mid \ldots \\
\beta &::= r \mid k \mid s \\
\gamma &::= \beta(\alpha) \\
\delta &::= (\gamma, \gamma) \\
\epsilon &::= (\delta, \delta, \ldots) \\
\phi &::= \gamma \mid \delta \mid \epsilon
\end{aligned}
$$

Language with influence and XOR relations

Language

Mirfak

Domain

Fragments are partitioned onto requirements, domain knowledge, and specifications, $\mathrm{F} = \mathrm{c.r} \cup \mathrm{c.k} \cup \mathrm{c.s}$ and $\mathrm{c.r} \cap \mathrm{c.k} \cap \mathrm{c.s} = \emptyset$. Positive and negative influences are over Fragments, $\mathrm{r.inf.pos} \subseteq \mathrm{F} \times \mathrm{F}$, $\mathrm{r.inf.neg} \subseteq \mathrm{F} \times \mathrm{F}$. r.xor is over $n > 1$ influence relation instances of the same type,

$$\mathrm{r.xor} \subseteq (\mathrm{r.inf.pos}^n) \cup (\mathrm{r.inf.neg}^n).$$

Language with influence and XOR relations

Language

Mirfak

Mapping

α symbols denote Fragments, and $r(\alpha)$ denote requirements $\mathcal{D}(r(\alpha)) \in$ c.r, $k(\alpha)$ denote domain knowledge $\mathcal{D}(k(\alpha)) \in$ c.k, and $s(\alpha)$ denote specifications, $\mathcal{D}(s(\alpha)) \in$ c.s. δ denote positive and negative influence relation instances, $\mathcal{D}(\delta) \in$ r.inf.pos $\cup$ r.inf.neg. ϵ symbols denote r.xor instances, $\mathcal{D}(\epsilon) \in$ r.xor.

Language with influence and XOR relations

Language
Mirfak
Language Services s.IsAlt

Alternative Composites

Exercise 18: Compose Fragments and relation instances into wholes

Choose any language defined so far.

- What do you need to add to that language, in order to represent that some Fragments and, or relation instances are parts of some whole?
- Would you do it with one or more new relations? If yes, which?
- Do you need one or more new categories? If yes, which?

How to form a Composite?

Relation

Fragment part of Composite (r.po)

Domain & Dimension

r.po $\subseteq$ F $\times$ C, where F is a set of Fragments and C is a set of Composites.

Properties

reflexive, antisymmetric, and transitive.

Reading

$(x, c) \in$ r.po reads "x is part of c".

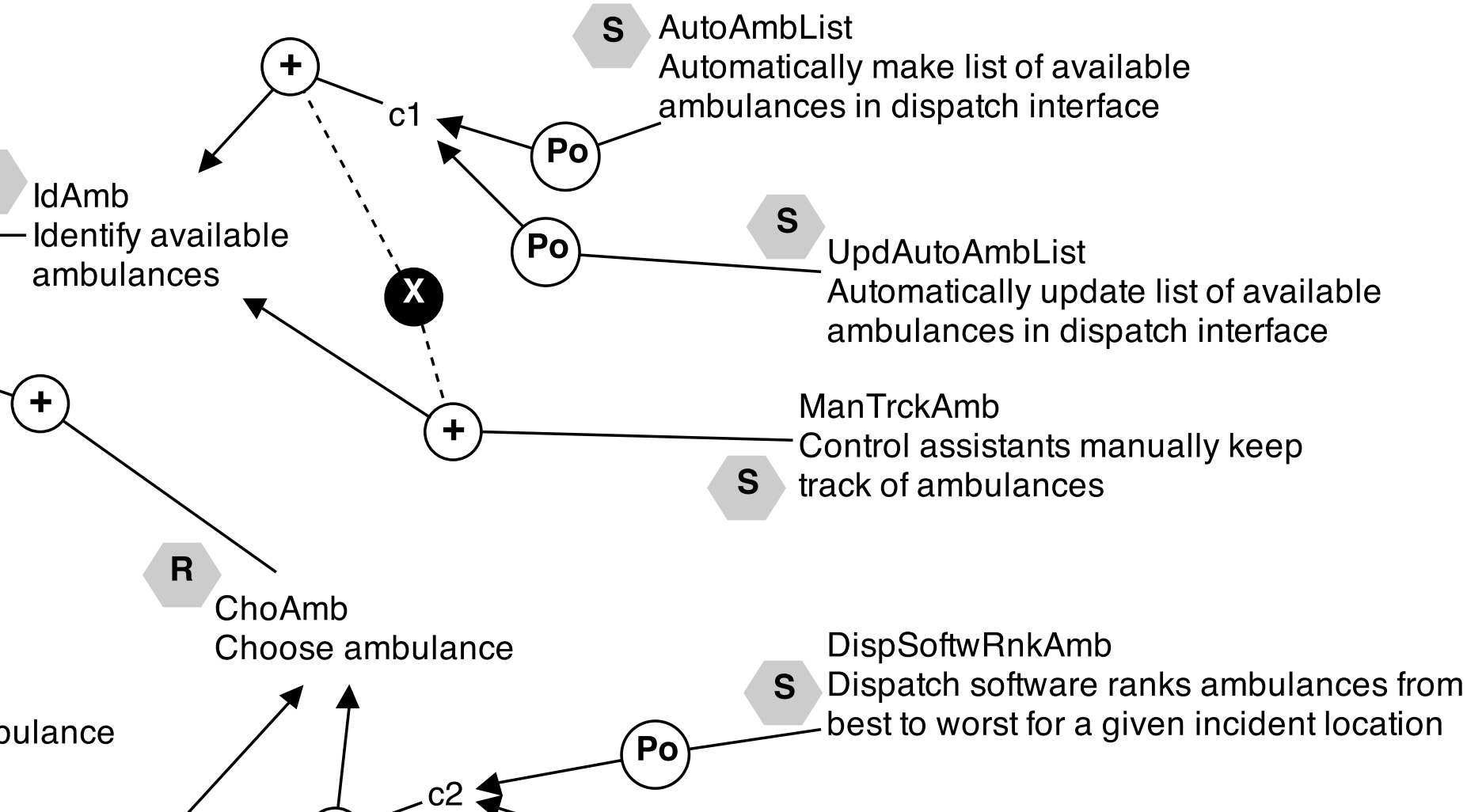
Language Services

- **s.IsPartOf**: Is x part of c? : Yes, if there is $(x, c) \in$ r.po.
- **s.HasParts**: Does x have parts? : Yes, if there is $(y, x) \in$ r.po.

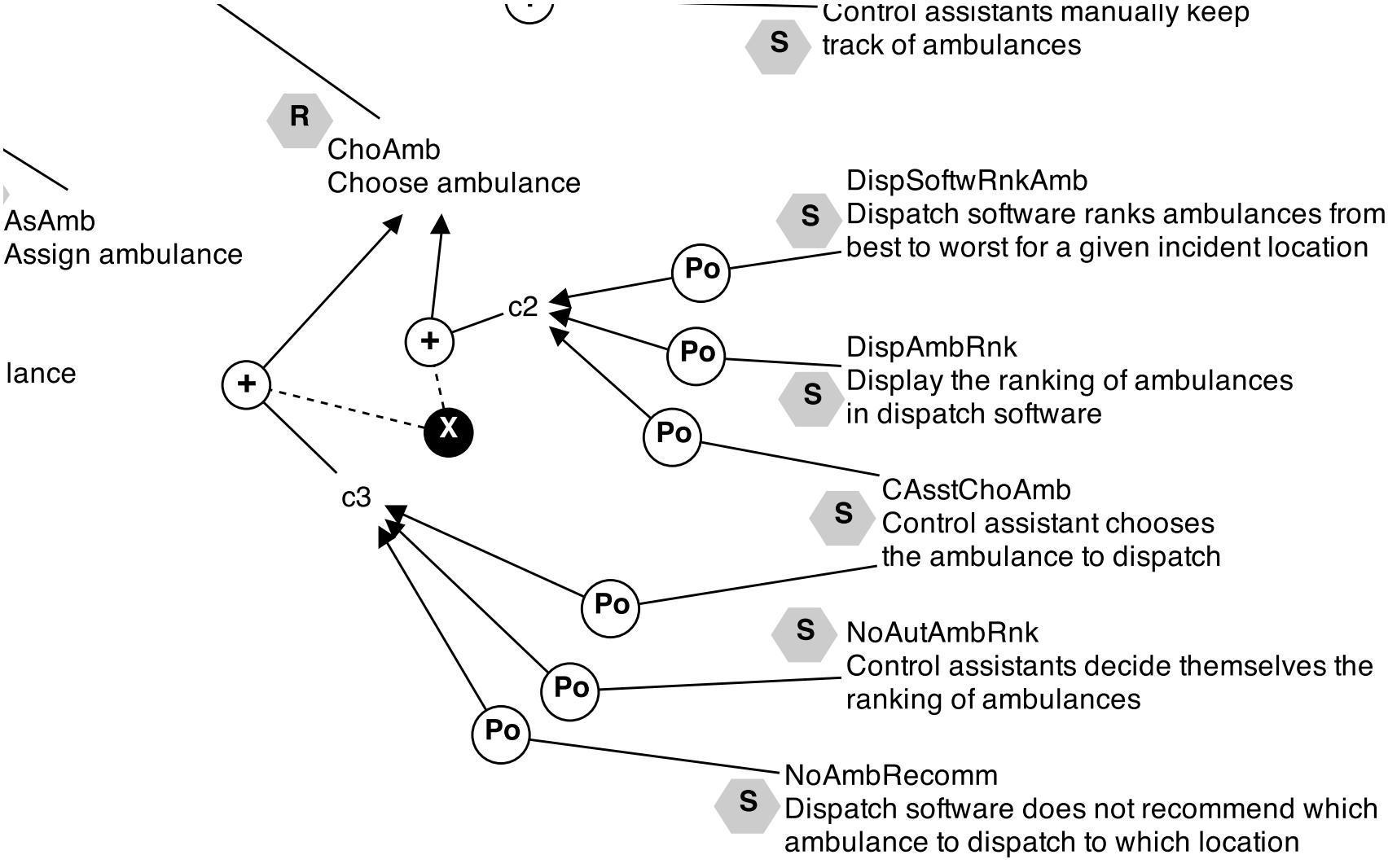
What is a Composite?

Category
Composite (c.cp)
Domain c.Composite $\subseteq F$, where F is a set of Fragments.
Membership conditions There is $y \in F$ such that $(y, x) \in$ r.po.
Reading $x \in$ c.cp reads "x is a Composite".
Language Services s.HasParts.

Example: A model with Composites



Example: A model with Composites



Combinations

Exercise 19 : Define a procedure which finds all Combinations in a L.Mirfak

Define a new category in L.Mirfak for Combinations. Define a procedure which takes a model in L.Mirfak and returns the set of all Combinations in that model.

What is a Combination?

Category

Combination (c.cb)

Domain

c.Combination $\subseteq R$, where $\wp(R)$ is a set of sets of relation instances.

Membership conditions

If $W \in$ c.cb , then there is for every $w \in W$ an instance $r \in$ r.xor, such that w is one of the Alternatives in r, and no member of $W \setminus \{w\}$ is also in r.

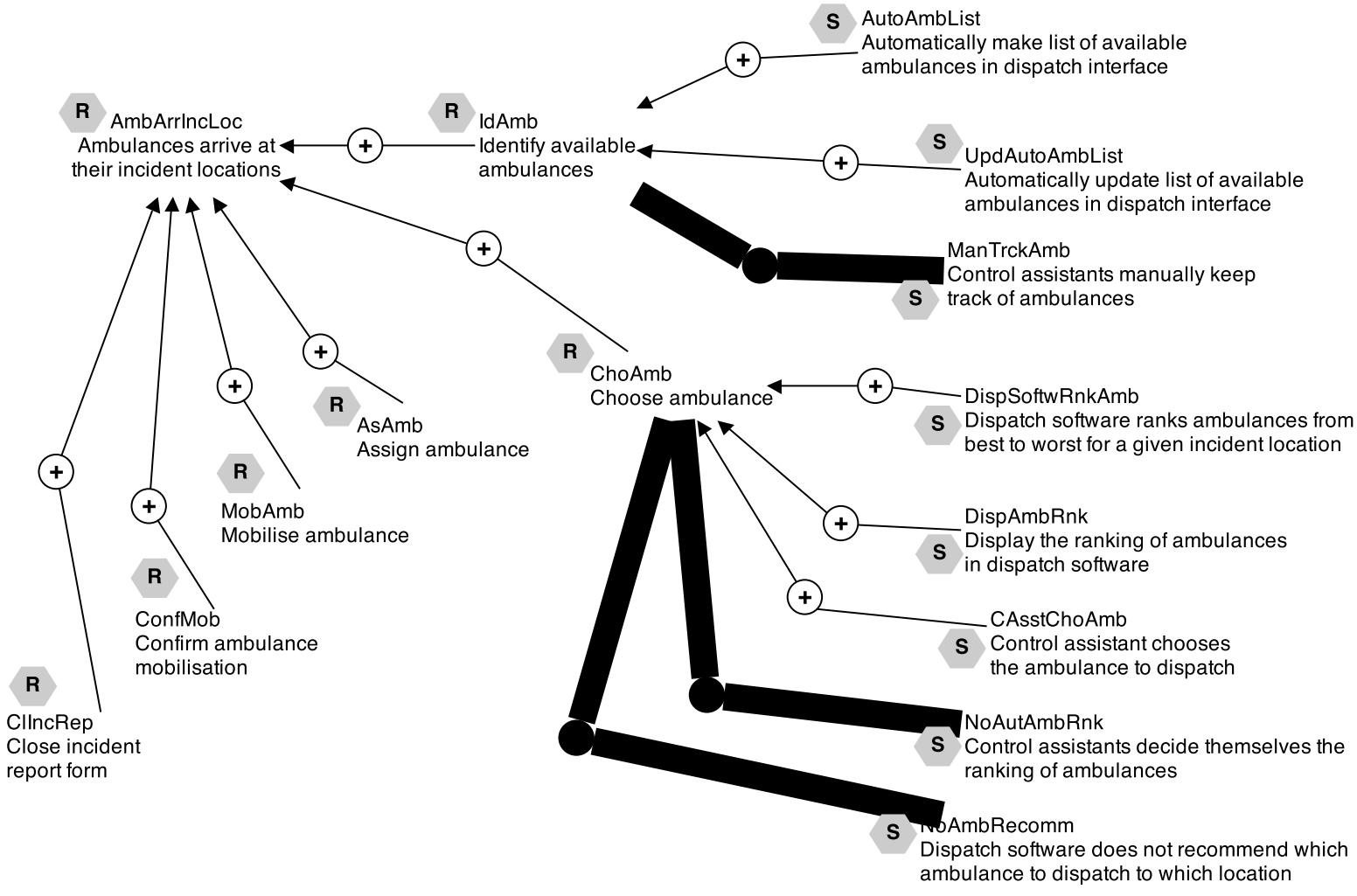
Reading

$W \in$ c.cb reads "W is a Combination".

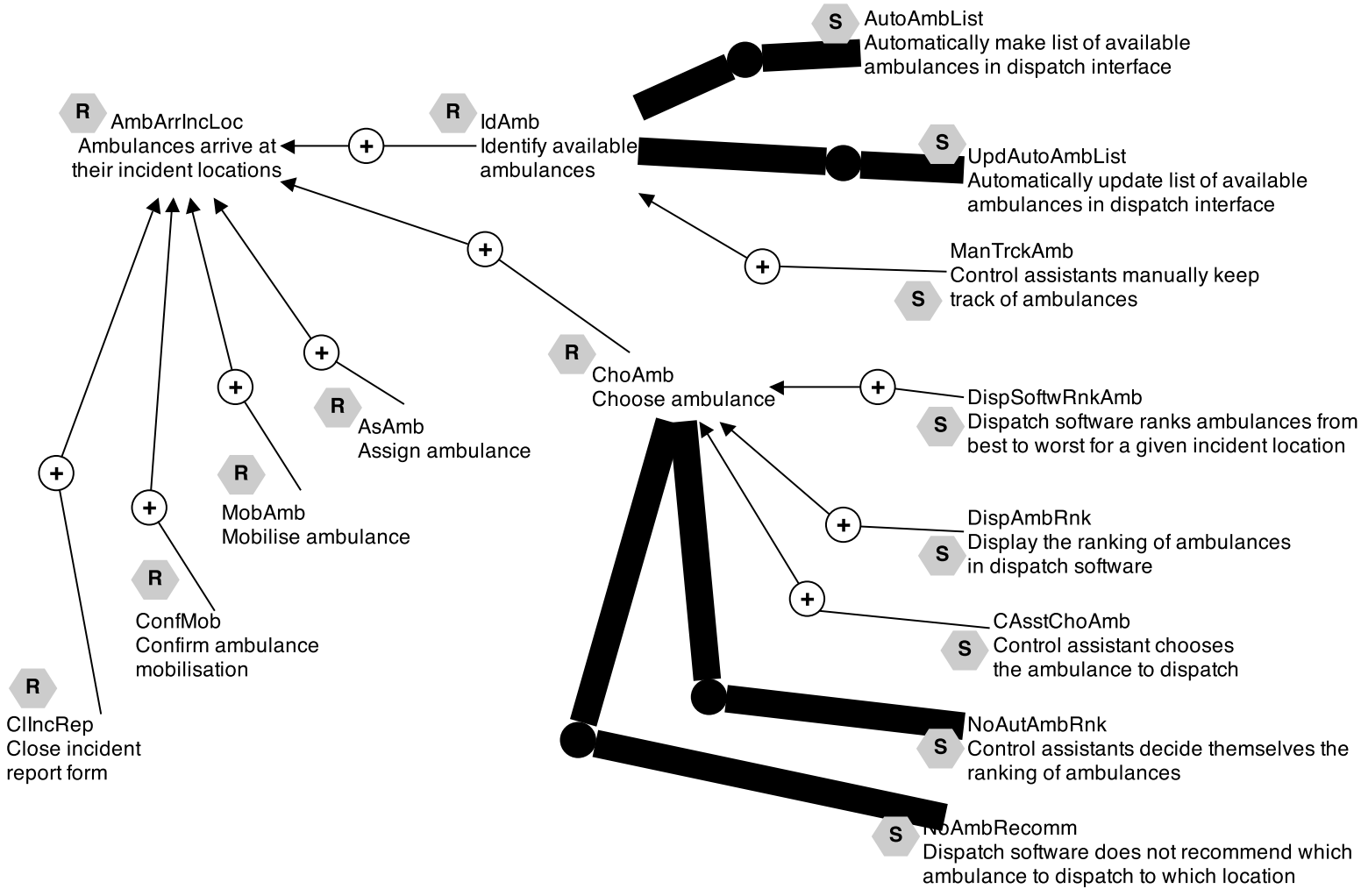
Language Services

- **s.IsCombination**: Is x a Combination? : Yes, if $x \in$ c.cb.

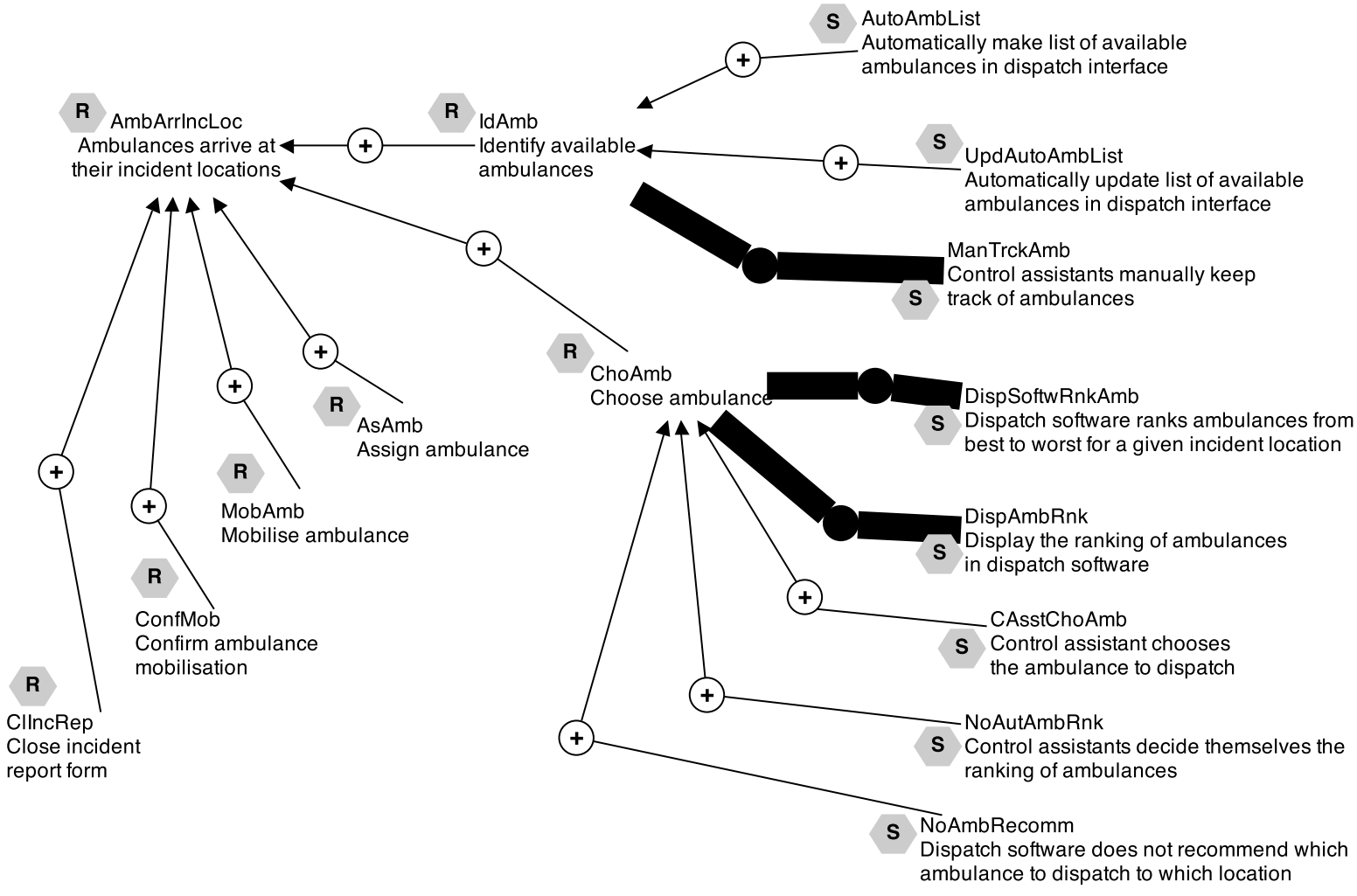
Example: Combination a



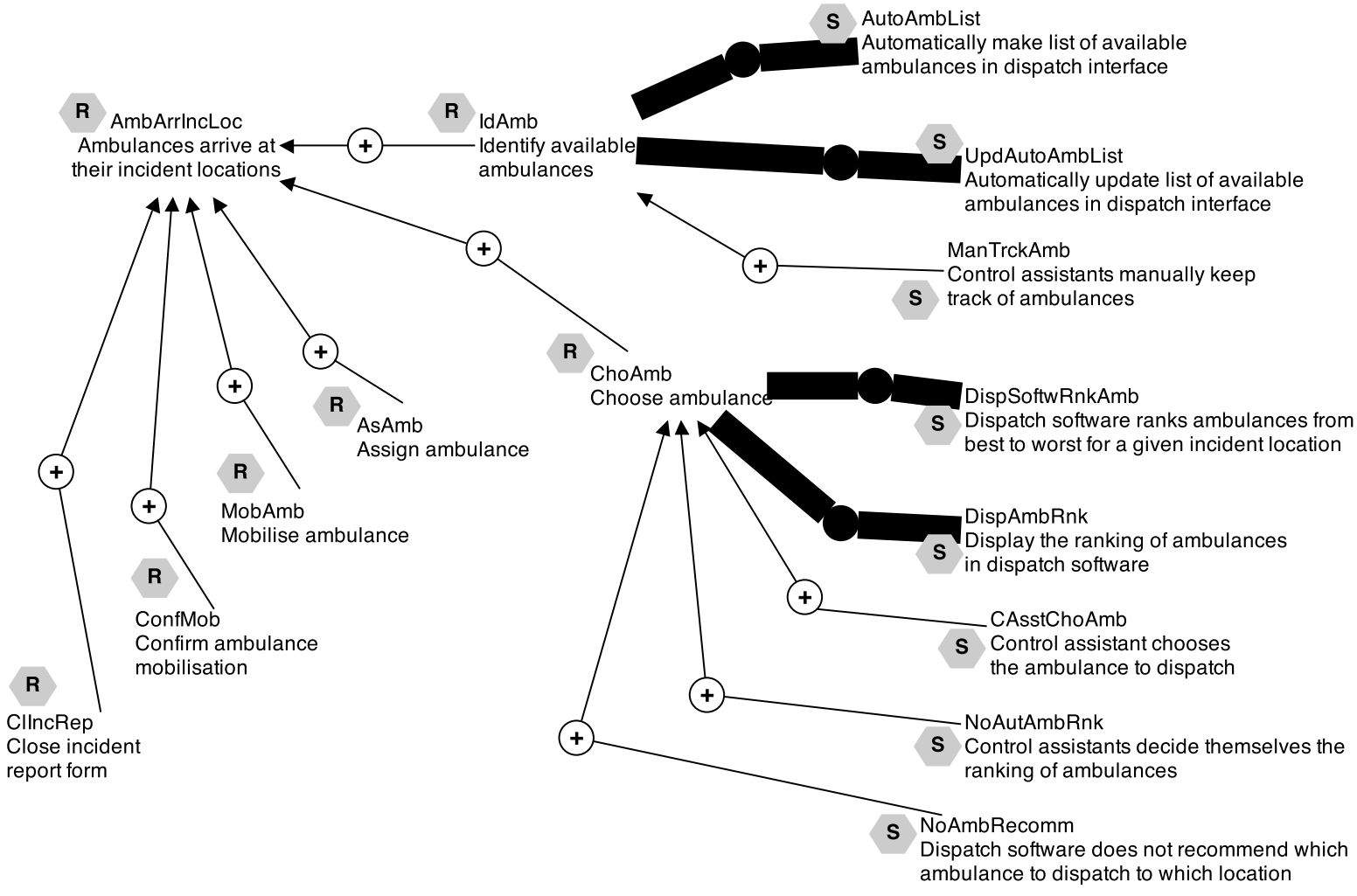
Example: Combination b



Example: Combination c



Example: Combination d



Sketch of a procedure to find Combinations

1. Take one r.xor instance,
2. Generate as many copies of the model, as there are Alternatives in the relation instance,
3. For each copy, if it has no more r.xor instances, then it is a Combination, otherwise, return to step 1 with that copy.

Formal Theories

Outline

1. Why map models to formal theories?
2. How to map models to formal theories?
3. What are the risks of doing this?

Formal theories?

A “formal theory” is a set of formal sentences.

Formal sentence is a formula without free variables.

It is a formula in some specific formal logic.

Why map models to formal theories?

To provide Language Services that you can't otherwise.

For example:

Language Service

DRPSol: Given a model *M* which includes an instance *P* of DRP, is the part *S* of that model a solution to the problem instance?

Exercise 20: Map models to propositional formal theories

- *What do you need to add to a language which has positive and negative influence relations, in order to map its models to propositional formal theories?*
- *Can the same model be mapped to different theories?*
- *If yes, then why would you map it to one of these theories and not another?*

A function that maps influence relations to a theory

Function

Map positive influences to implications, negative influences to inconsistency
(f.map.infl.impl)

Input

Model M.

Do

1. Let Δ be an empty set.
2. For every Fragment x in M:
   - 2.1 Let $\{(p_1, x), \ldots, (p_n, x)\}$ be the set of all positive influences to x in M.
   - 2.2 Let $\{(q_1, x), \ldots, (q_m, x)\}$ be the set of all negative influences to x in M.
   - 2.3 Add the following sentences to Δ:

$$\begin{aligned} p_1 \wedge \ldots \wedge p_n &\rightarrow x, \\ q_1 \wedge \ldots \wedge q_m \wedge x &\rightarrow \bot. \end{aligned}$$

A function that maps influence relations to a theory

Function

Map positive influences to implications, negative influences to inconsistency
(f.map.infl.impl)

Input

Model M.

Do

...

Output

Set Δ of propositional logic sentences.

Language Services

- **s.WhCPLTh**: What CPL theory corresponds to positive and negative influences over Fragments in M? : Δ.

Example: A formal theory

Suppose that you are given the following CPL formulas:

$$
\begin{aligned}
F_{c.k} &= \{ \quad \mathsf{SwtchCal}, \mathsf{NoDropCal} \ \} \\
F_{c.s} &= \{ \quad \mathsf{SrcMap}, \mathsf{IncLocVisSw}, \mathsf{UpdOpIncLoc}, \mathsf{FilSwIncRep} \ \} \\
F_{c.r} &= \{ \quad \mathsf{AddRepEm} \ \} \\
\Delta &= \{ \quad \mathsf{FilIncRep} \wedge \mathsf{ChkDblLoc} \wedge \mathsf{IdIncLoc} \wedge \mathsf{RecEmCal} \wedge \mathsf{NoDropCal} \wedge \mathsf{SwtchCal} \\
&\qquad \rightarrow \mathsf{AddRepEm}, \\
&\qquad \mathsf{SwtchCal} \wedge \mathsf{NoDropCal} \rightarrow \mathsf{RecEmCal}, \\
&\qquad \mathsf{SrcMap} \rightarrow \mathsf{IdIncLoc}, \\
&\qquad \mathsf{UpdOpIncLoc} \wedge \mathsf{IncLocVisSw} \rightarrow \mathsf{SwIdDuplCal}, \\
&\qquad \mathsf{SwIdDuplCal} \rightarrow \mathsf{ChkDblLoc}, \\
&\qquad \mathsf{FillSwIncRep} \rightarrow \mathsf{FillIncRep} \ \}.
\end{aligned}
$$

Example: Questions

- What Fragments and relations are in a model M, which includes only the Fragments that correspond to atomic propositions in $F_{c.k}$, $F_{c.s}$, and $F_{c.r}$ above, and has influences which mapped to those in Δ above, via f.map.inf.impl?
- Is there a Default RP problem instance in M?
- Is S given above a solution to that Default RP instance in M?

What are the risks of mapping to formal theories?

Getting misleading conclusions if you are not careful.

For example,

$$K, \Delta \vdash x,$$

will be satisfied, for any x, if $K, \Delta \vdash \bot$.

Solving Default RPs with these languages

Suppose the language only has

- Fragments,
- Positive and negative influence relations,
- f.map.infl.impl.

Can it solve the Default RP?

Which problem can that language solve?

The following problem:

Given:

- a set R of requirements Fragments, and
- a set K of domain knowledge Fragments,

Define a set S of specification Fragments, which satisfies

- the provability condition $K, S, \Delta \vdash R$, and
- the consistency condition $K, S, \Delta \nvdash \bot$.

Next steps

For all other topics of this tutorial, go to

`http://www.jureta.net/requirements-modelling-languages`

and download the text which covers additional topics, including:

- Valuation
- Uncertainty and Probability
- Preferences
- Problem Classes