A Tool for Automated Reasoning about Traces Based on Configurable Formal Semantics

Ferhat Erata[1,4] Arda Goknil[2] Geylani Kardas[3]
Bedir Tekinerdogan[4]

[1]UNIT Information Technologies R&D Ltd., Izmir, Turkey
[2]SnT Centre for Security, Reliability and Trust, University of Luxembourg
[3]International Computer Institute, Ege University, Izmir, Turkey
[4]Information Technology Group, Wageningen University, The Netherlands

11th Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering

Exploitations

ITEA-ModelWriter: Synchronized Document Engineering Platform

https://itea3.org/project/modelwriter.html

ITEA-ASSUME: Affordable Safe & Secure Mobility Evolution

https://itea3.org/project/assume.html

ITEA3

EUREKA Σ!
innovation across borders

Source codes, datasets and screencasts are available at:

https://modelwriter.github.io/Tarski/

Outline

1. Motivation & Challenges
   - Motivation
   - Challenges
2. Industrial Use Case
   - Some Requirements and Code Fragments in ECAS
   - Example Inferred and Inconsistent Traces in ECAS
3. Overview of the Tool
   - Specify Project-Specific Trace Types and Semantics
   - Assign Traces within the Project Artifacts
   - Assign Traces between the Project Artifacts
   - Reasoning about Traceability
4. Evaluation and Lessons Learned
   - Evaluation
   - Lessons Learned

Motivation

What is Traceability?

Traceability can be defined as the degree to which a relationship can be established among work products (aka. artefacts) of the development process.

What is case-based or project-based traceability configuration?

Rigorously specification the semantics of traceability elements.

Why is Reasoning about Traceability important?

Richer and precise automated traceability analysis.
Compliance and Certification in automotive and aviation industries.

Challenges of Traceability in Industry

Semantically meaningful traceability

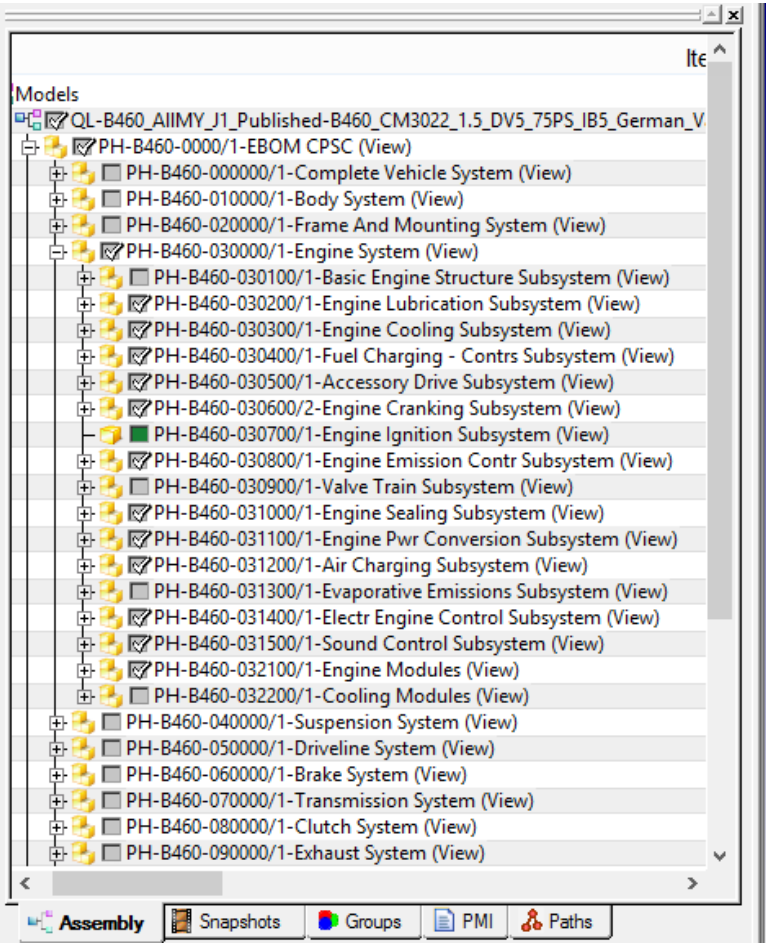
- traceability relations should have a rich semantic (meaning) instead of being simple bi-directional referential relation

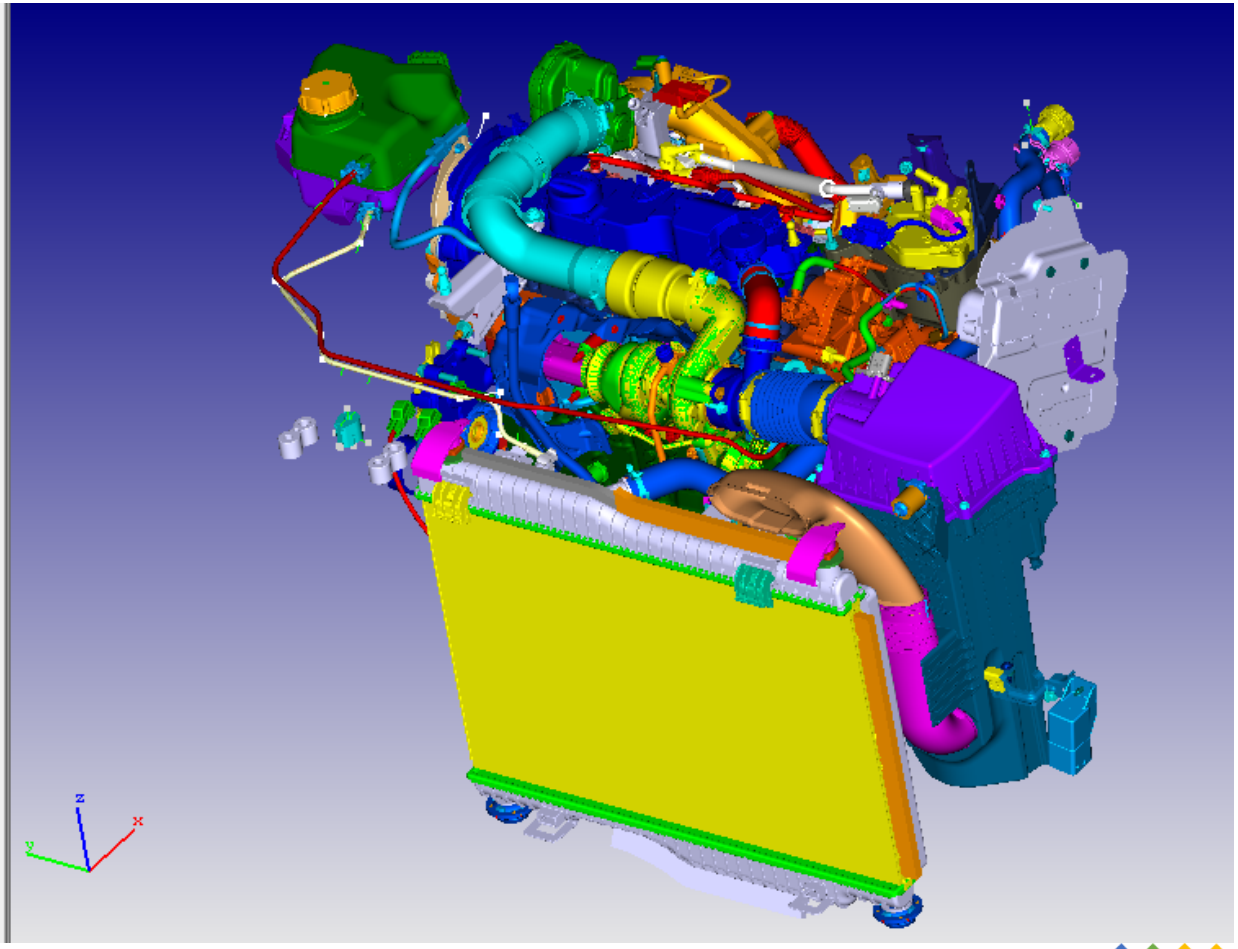
Configuration of traceability (possibly dynamically)

- Traceability Semantics is often statically defined in the tools.
- The semantics cannot be easily adapted for the needs of different projects.
- Different traceable elements and the relation types exist in industrial settings,
- Likewise, different traceability analysis scenarios exists. Several industries demands formal proofs of Traceability.

Ford-Otosan Motor Company

Electronically Controlled Air Suspension (ECAS) System

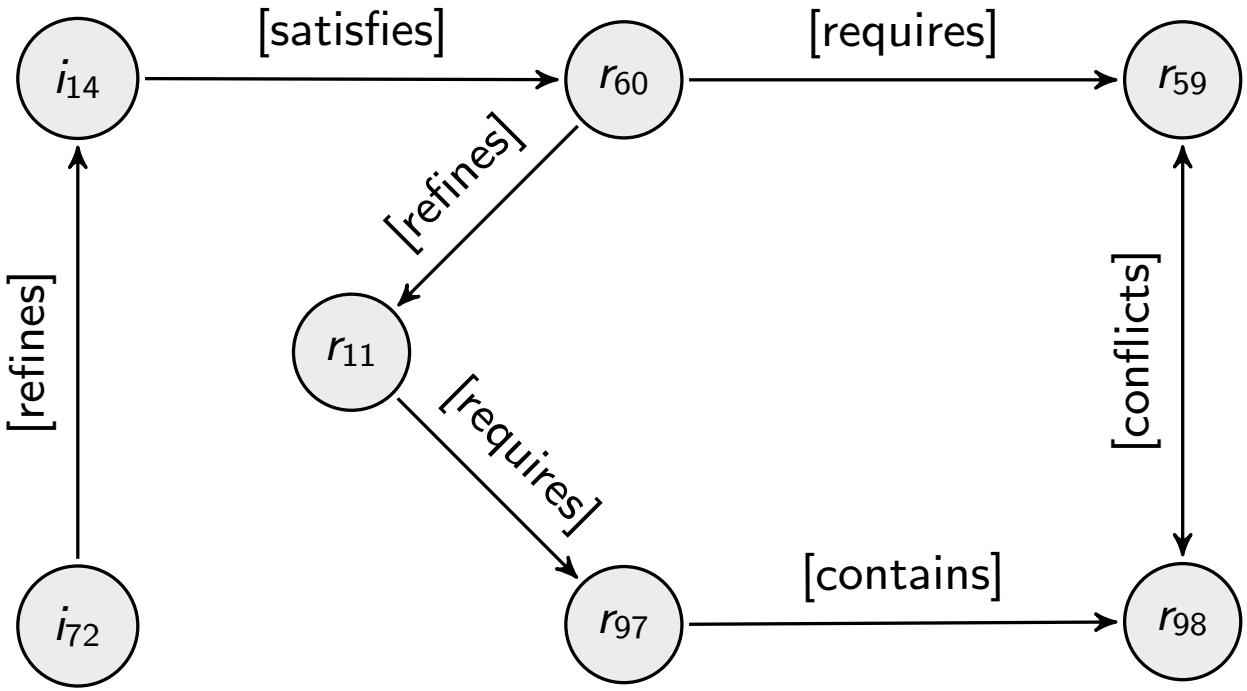




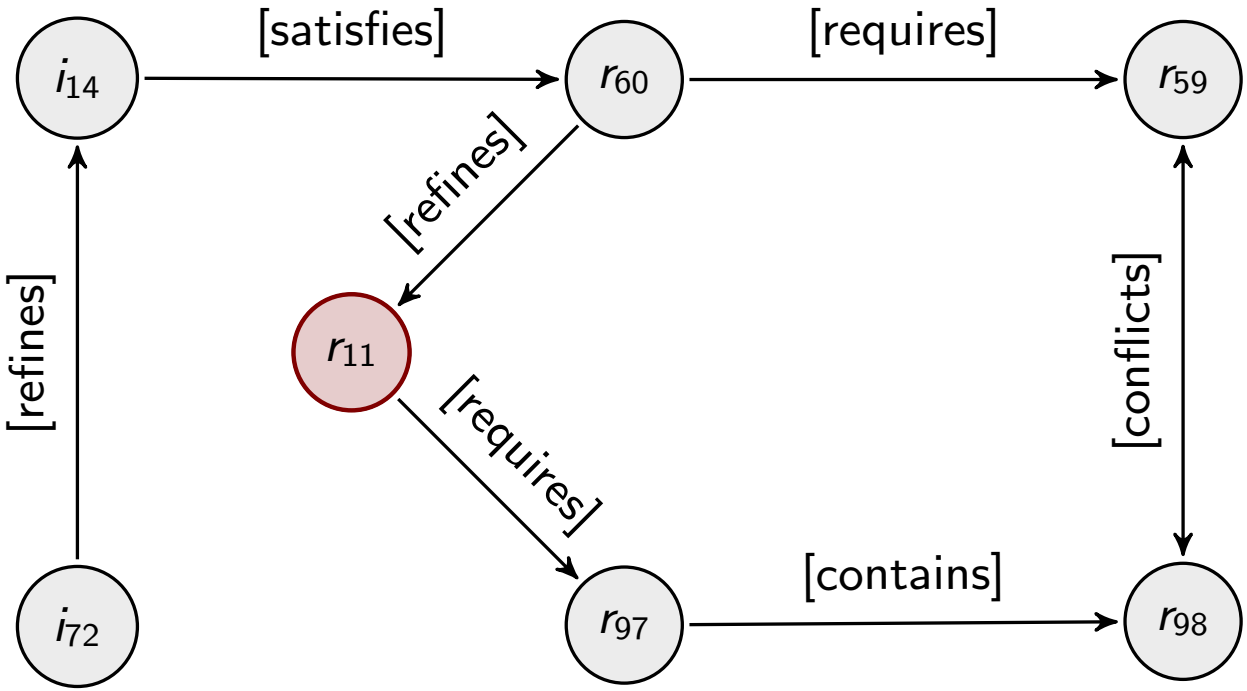
Some Requirements and Code Fragments in ECAS

Nr.	Requirements/Code Fragments
r_{11}	The system shall do height corrections using long and short term filtered height sensor signal.
r_{59}	The system shall always use height sensors in the range of 0-5V to avoid long term signal filtering.
r_{60}	The system shall do height corrections using long and short term filtered height sensor signal with 10ms interval.
r_{97}	The system shall filter height sensor signal in short term and long term for height corrections.
r_{98}	The system shall filter height sensor signal in long term for height corrections.
i_{14}	`vehicle::ecas::processHeightSensor::filterSignal`
i_{72}	`vehicle::ecas::processHeightSensor`

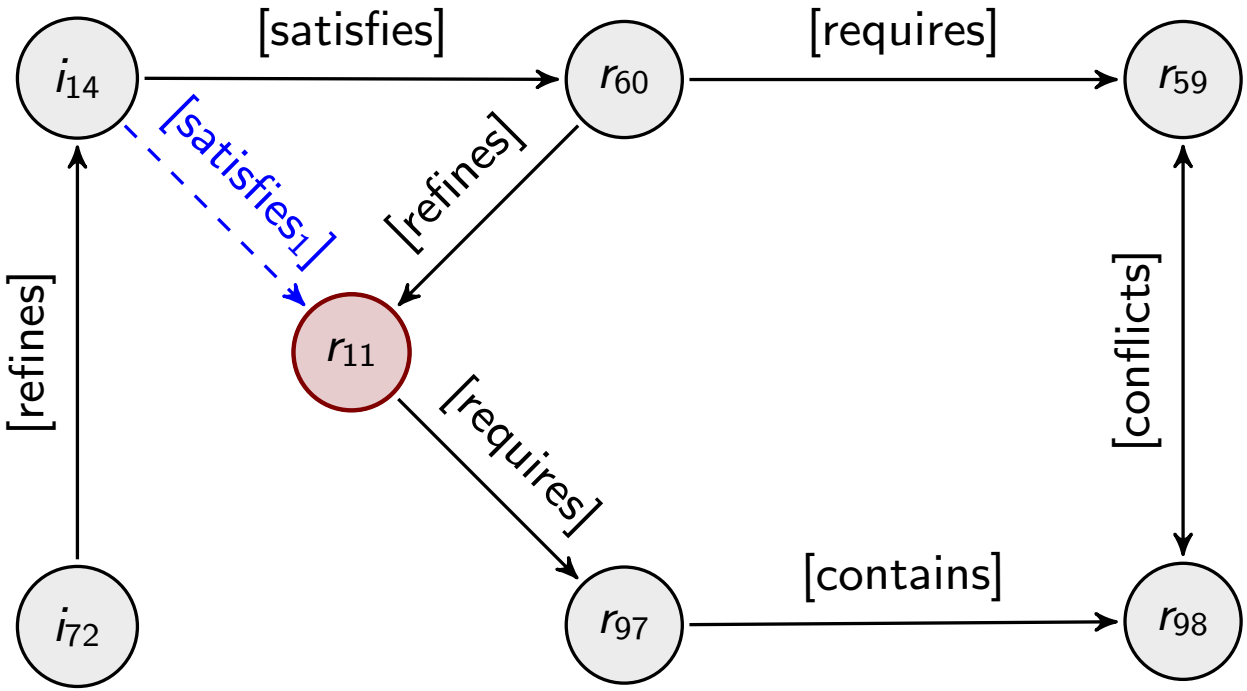
Example Inferred and Inconsistent Traces in ECAS



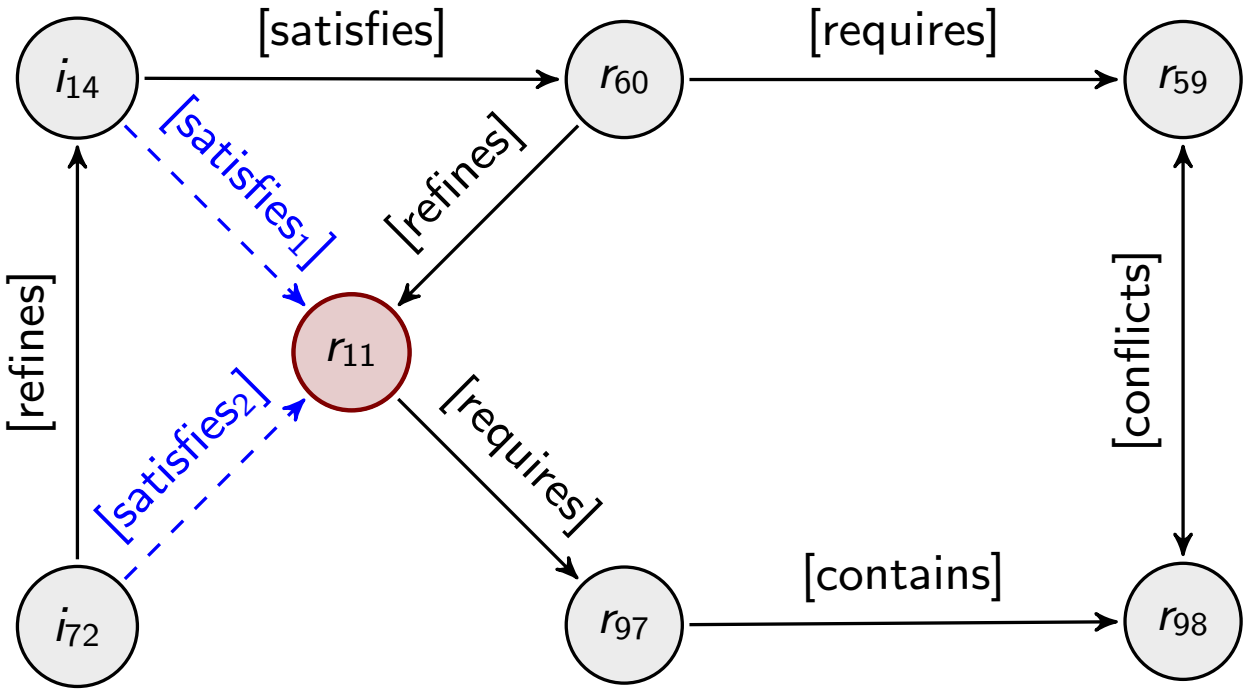
Which implementation artifacts satisfy r_{11}?



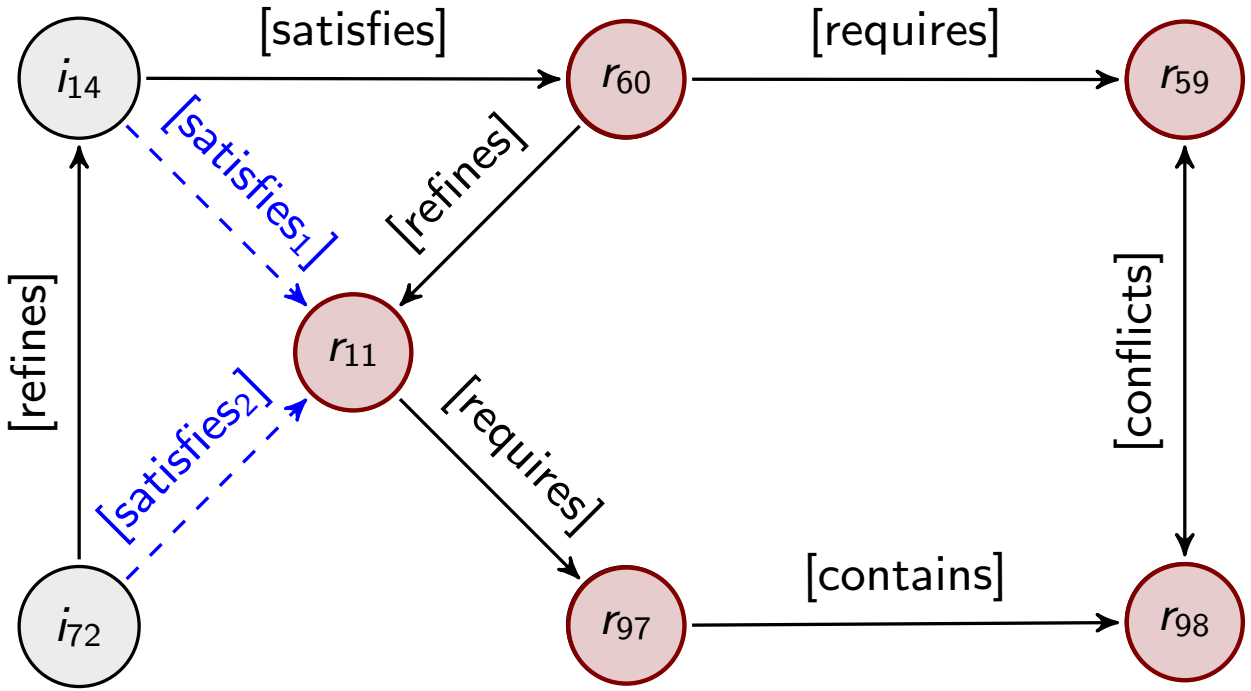
Which implementation artifacts satisfy r_{11}?



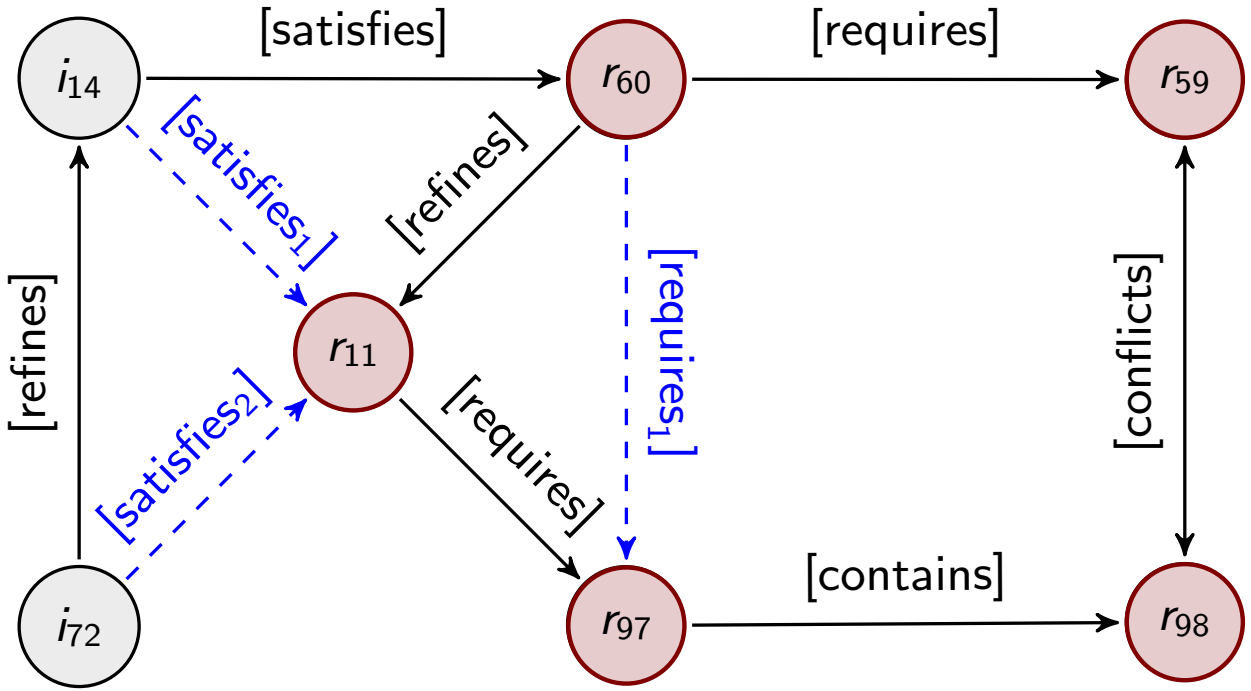
Which implementation artifacts satisfy r_{11}?



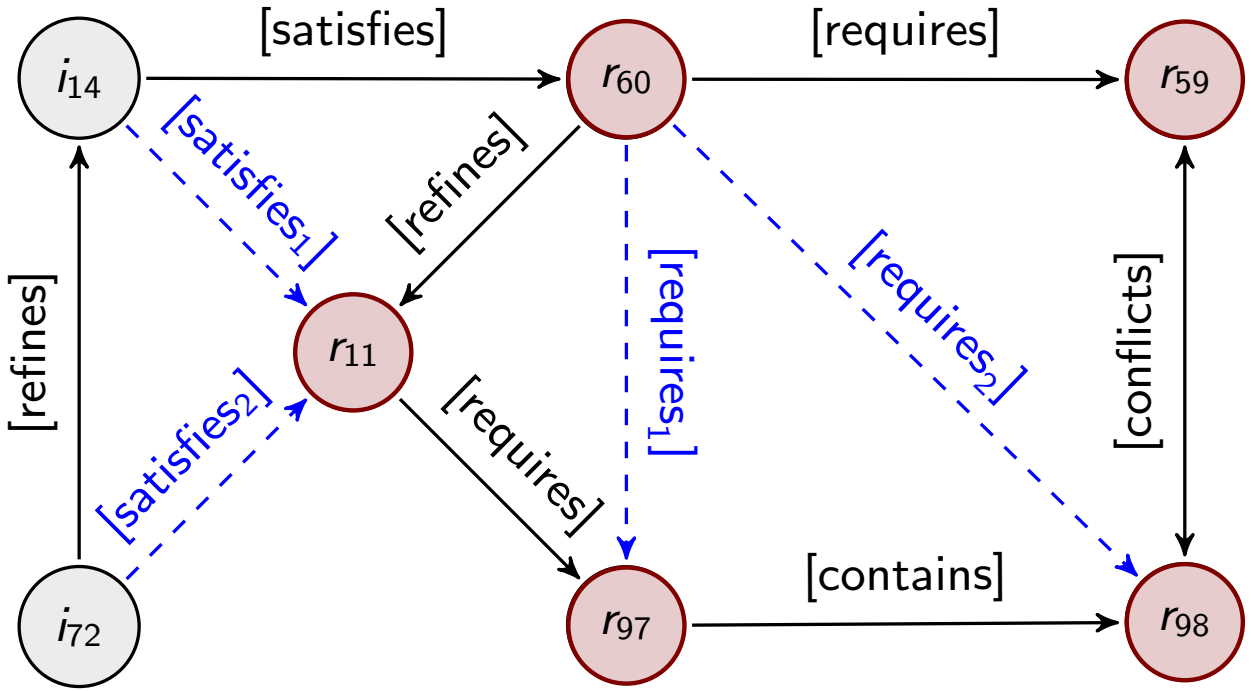
Is there any inconsistency among Requirements?



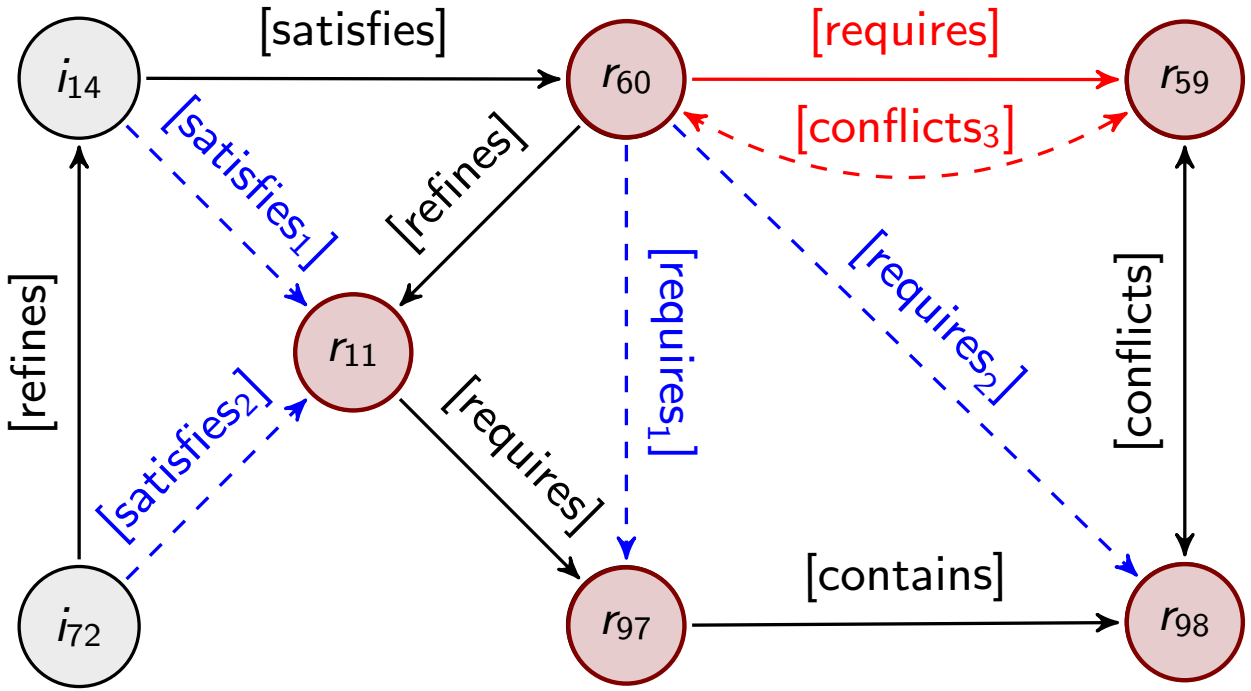
Is there any inconsistency among Requirements?



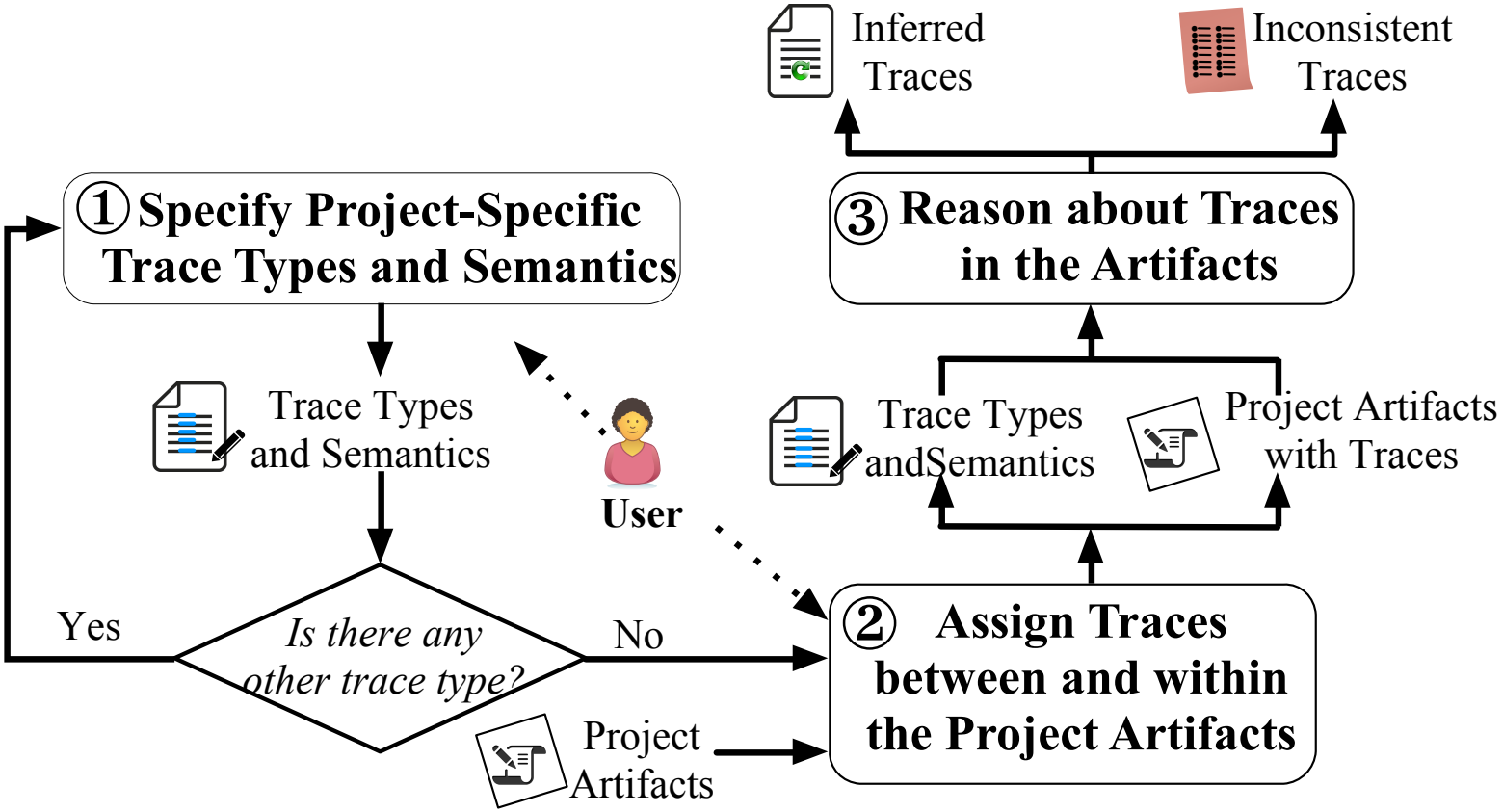
Is there any inconsistency among Requirements?



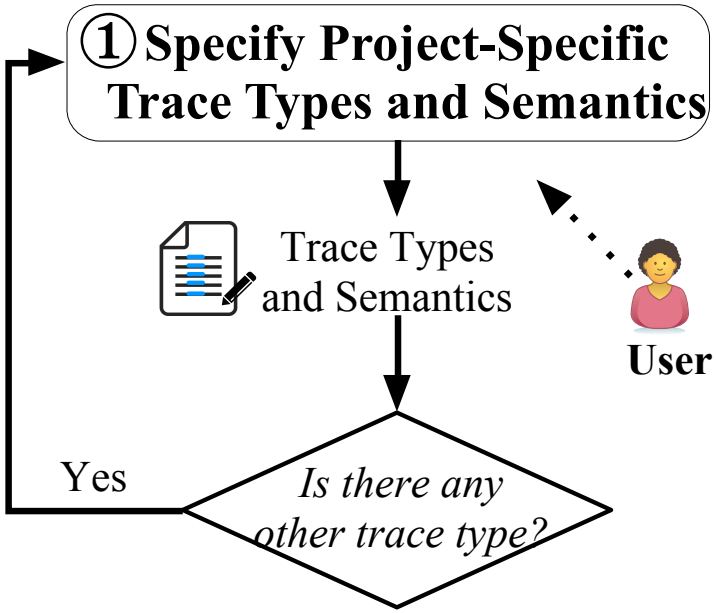
Is there any inconsistency among Requirements?



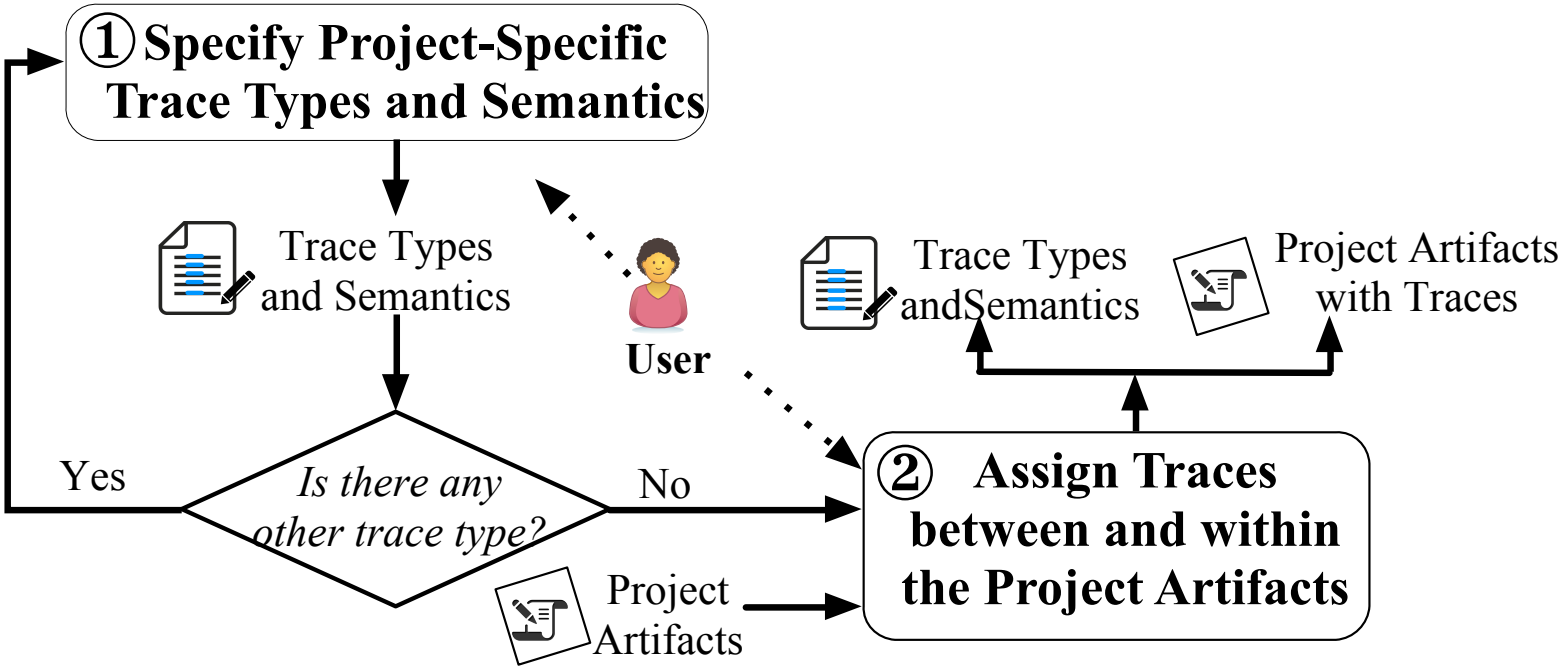
Overview of the Tool



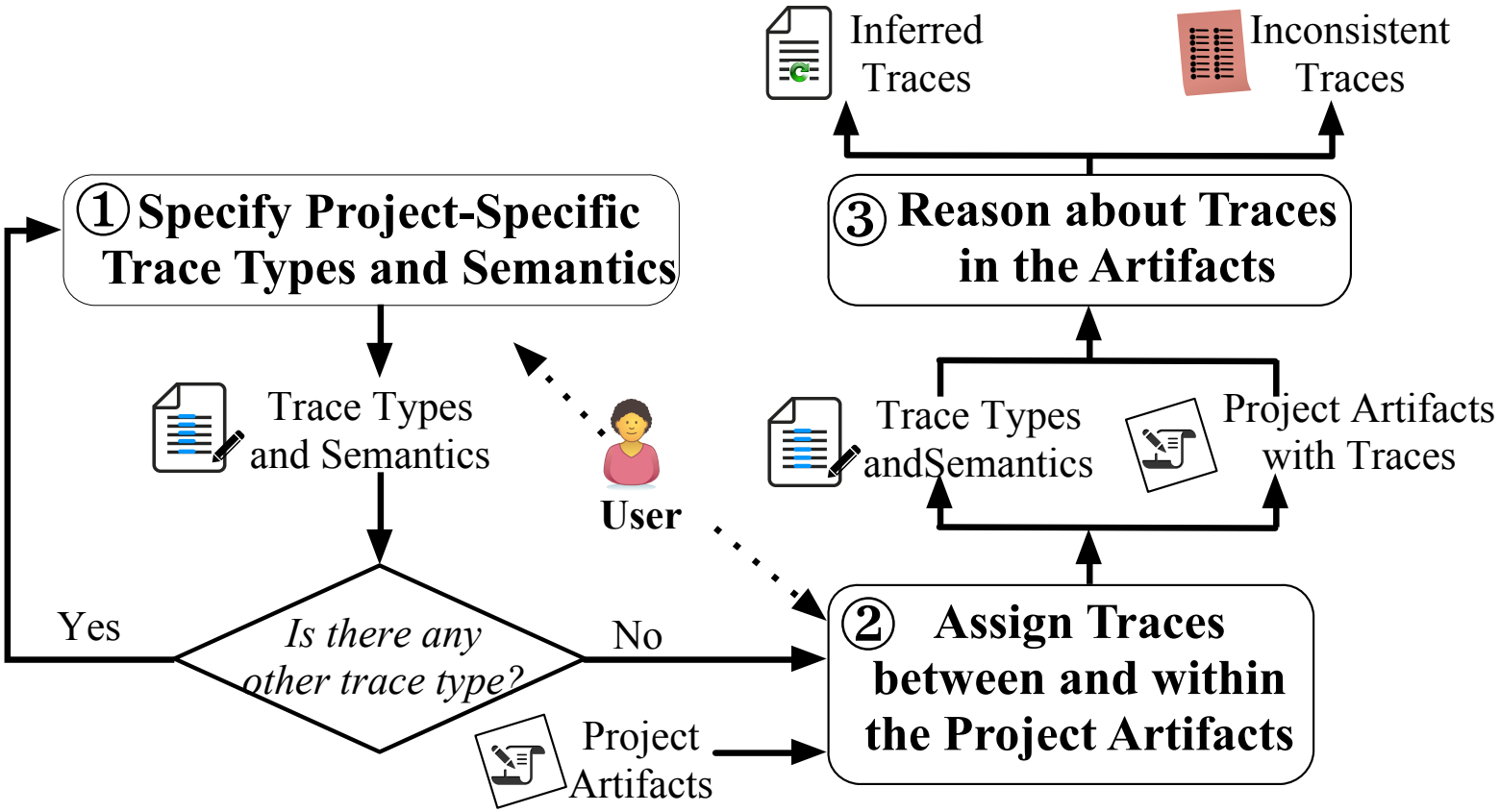
① Specify Project-Specific Trace Types and Semantics



② Assign Traces between and within the Project Artifacts



③ Reason about Traces in the Artifacts



① Specify Project-Specific Trace Types

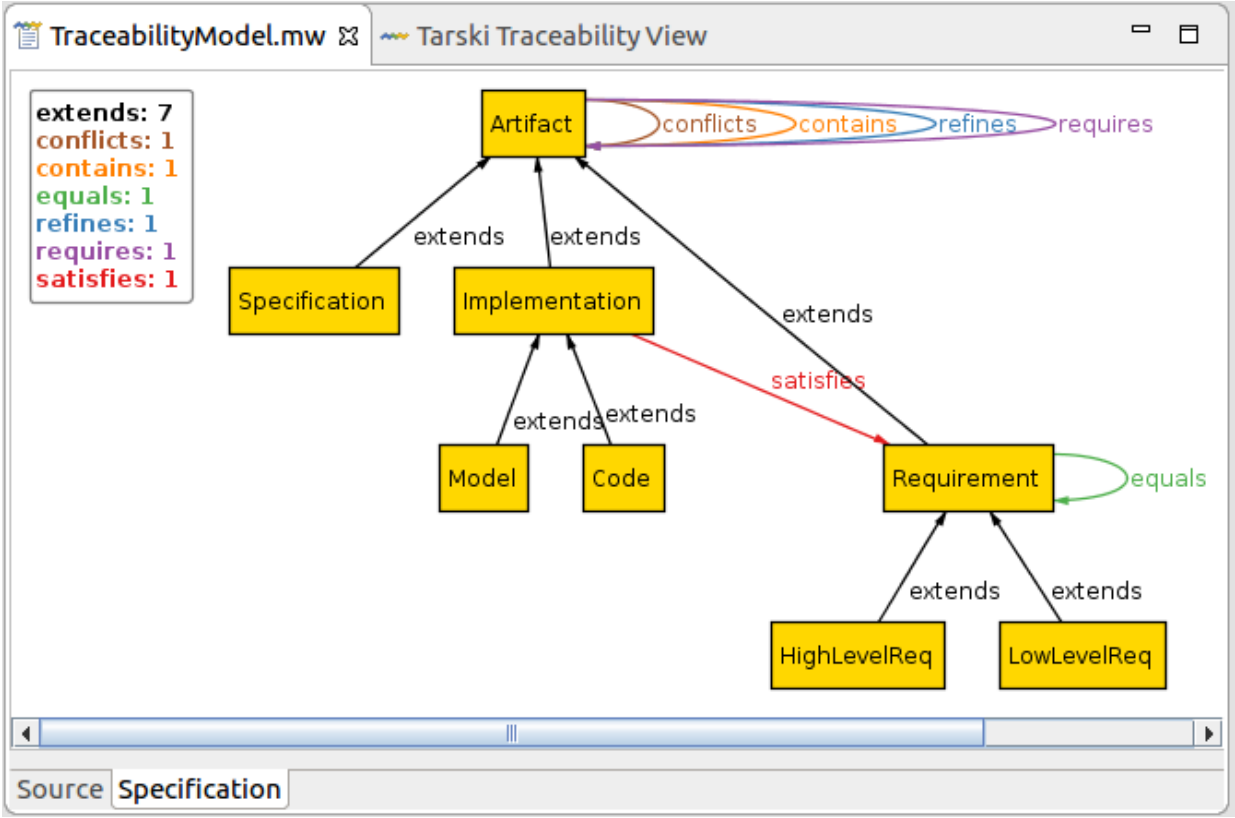
File Edit Navigate Search Tarski Project Run Window Help

*TraceabilityModel.mw

```
1 module TraceabilityModel
2 open util/relation
3
4 abstract sig Artifact {
5     requires: set Artifact, refines: set Artifact,
6     contains: set Artifact, conflicts: set Artifact}
7
8 -- Locate@Text
9 sig Specification extends Artifact {}
10
11 -- Locate@Text
12 sig Requirement extends Artifact { equals: set Requirement}
13
14 -- Locate@
15 sig LowLevelReq, HighLevelReq extends R { }
16
17 abstract sig Implementation extends Artifact {
18     satisfies: set Requirement}
19
20 -- Locate@Code
21 sig Code extends Implementation {}
22
23 -- Locate@Model
24 sig Model extends Implementation {}
```

Source Specification

① Trace Type Hierarchy of ECAS



① Specify Project-Specific Trace Semantics

File Edit Navigate Search Tarski Project Run Window Help

*TraceabilityModel.mw

```
25
26 fact {injective[contains, Artifact]}
27 fact {irreflexive[requires + refines + contains + conflicts]}
28 fact {antisymmetric[requires + refines + contains]}
29
30 -- Reason@Artifact.conflicts
31 fact {all a,b,c: Artifact |
32         b in a.(requires + refines + contains) and
33                     c in b.conflicts => c in a.conflicts
34     symmetric[conflicts]}
35
36 -- Reason@Implementation.satisfies
37 fact {all a,b,c : Artifact {
38         b in a.refines and c in b.satisfies =>
39                                     c in a.satisfies
40         b in a.refines and a in c.satisfies =>
41                                     b in c.satisfies }}
42 -- Reason@Artifact
43 fact {all a,b,c: Artifact {
44         b in a.requires and c in b.(refines + contains) and
45            c !in a.(refines + contains) => c in a.requires
46         b in a.(refines + contains) and c in b.requires and
47            c !in a.(refines + contains) => c in a.requires}}
48
fact {no conflicts & (requires + refines + satisfies + contair
```

Source | Specification

② Assign Traces between and within the Project Artifacts

TraceabilityModel.mw | Process Height Sensor Signal.v1.md

```
51 ## Process Height Sensor Signal/Voltage
52
53 The system should make height corrections by raising or lowering
   vehicle until ride height is met based on the height sensor signal.
54
55 ### REQ_1
56
57 Height sensor signal should be sampled & filtered 10 ms intervals.
```

Markdown Source | Preview

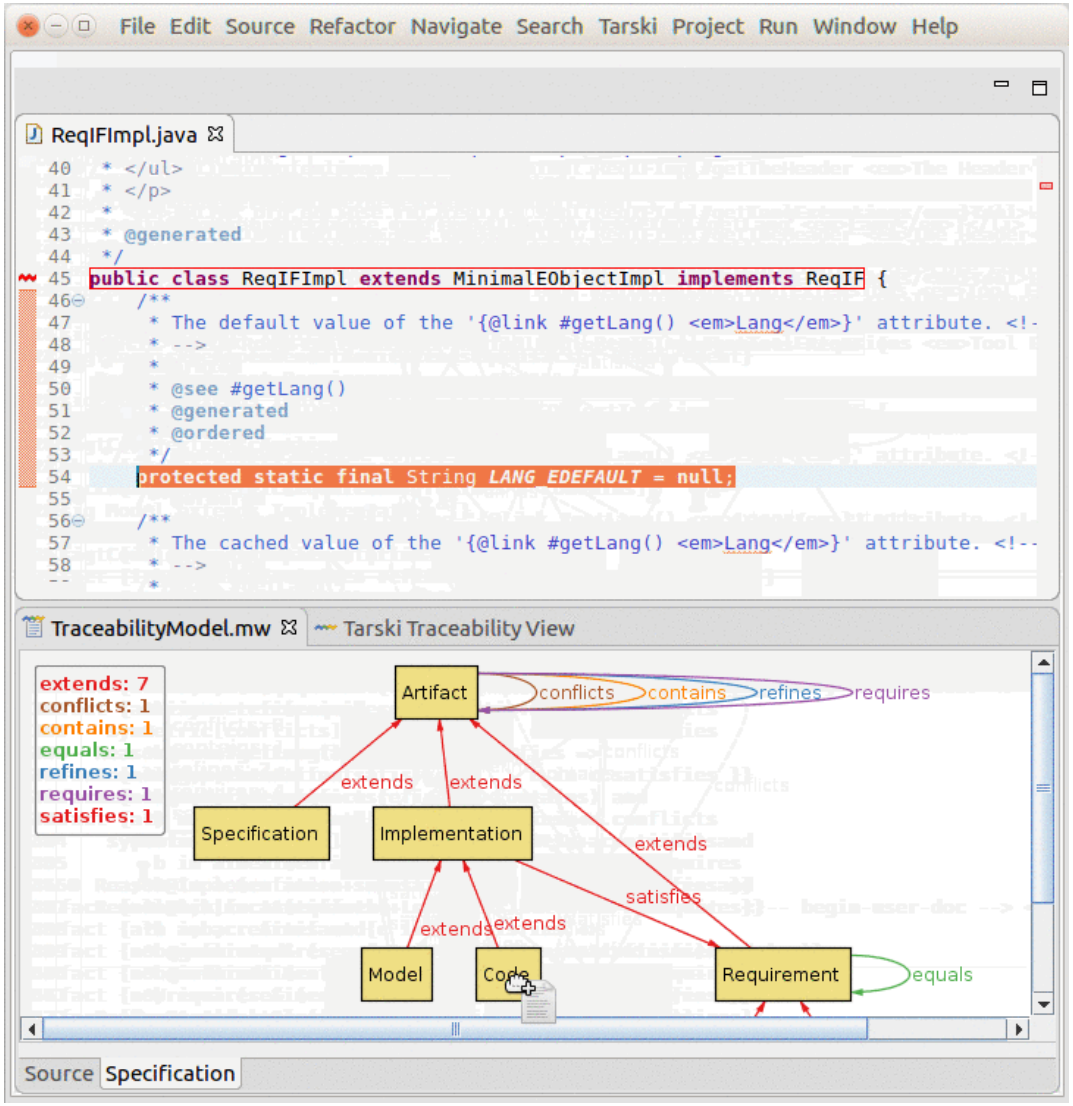
reqif10.ecore

- platform:/resource/eu.modelwriter.demonstration.requirements/reqif1(
  - reqif10
    - ReqIF
      - coreContent : ReqIFContent
      - toolExtensions : ReqIFToolExtension
    - ReqIFContent

reqif10.ecore | ReqIF.java

```
1 package org.eclipse.rmf.reqif10;
2
3 import org.eclipse.emf.common.util.EList;
4 import org.eclipse.emf.ecore.EObject;
5
6 public interface ReqIF extends EObject {
7
8     ReqIFHeader getTheHeader();
```

② Assign Traces within the Project Artifacts



② Assign Traces within the Project Artifacts

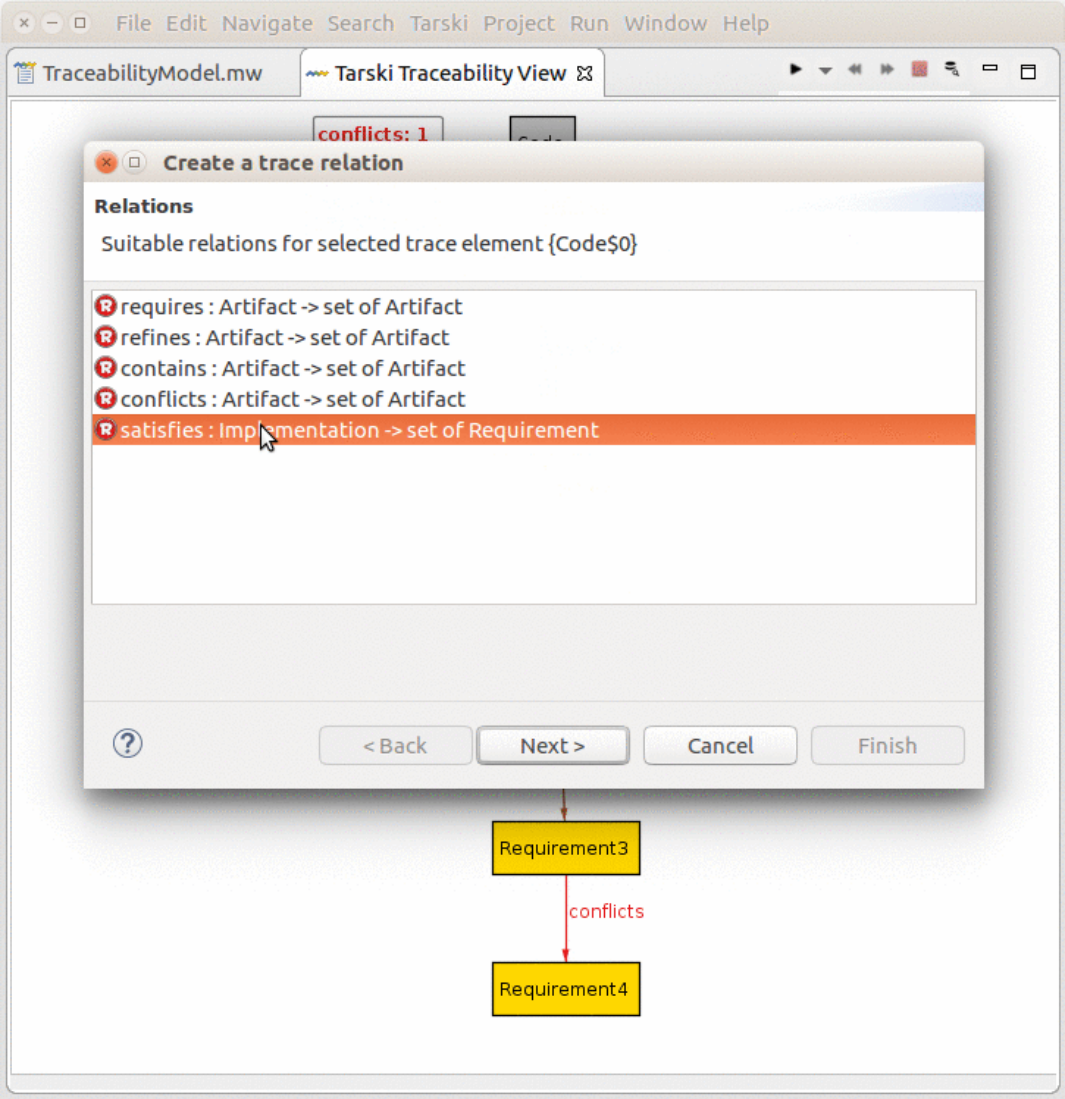
File Edit Navigate Search Tarski Project Run Window Help

Tarski Traceabi | ReqIFImpl.java | reqif10.ecore | Process Height

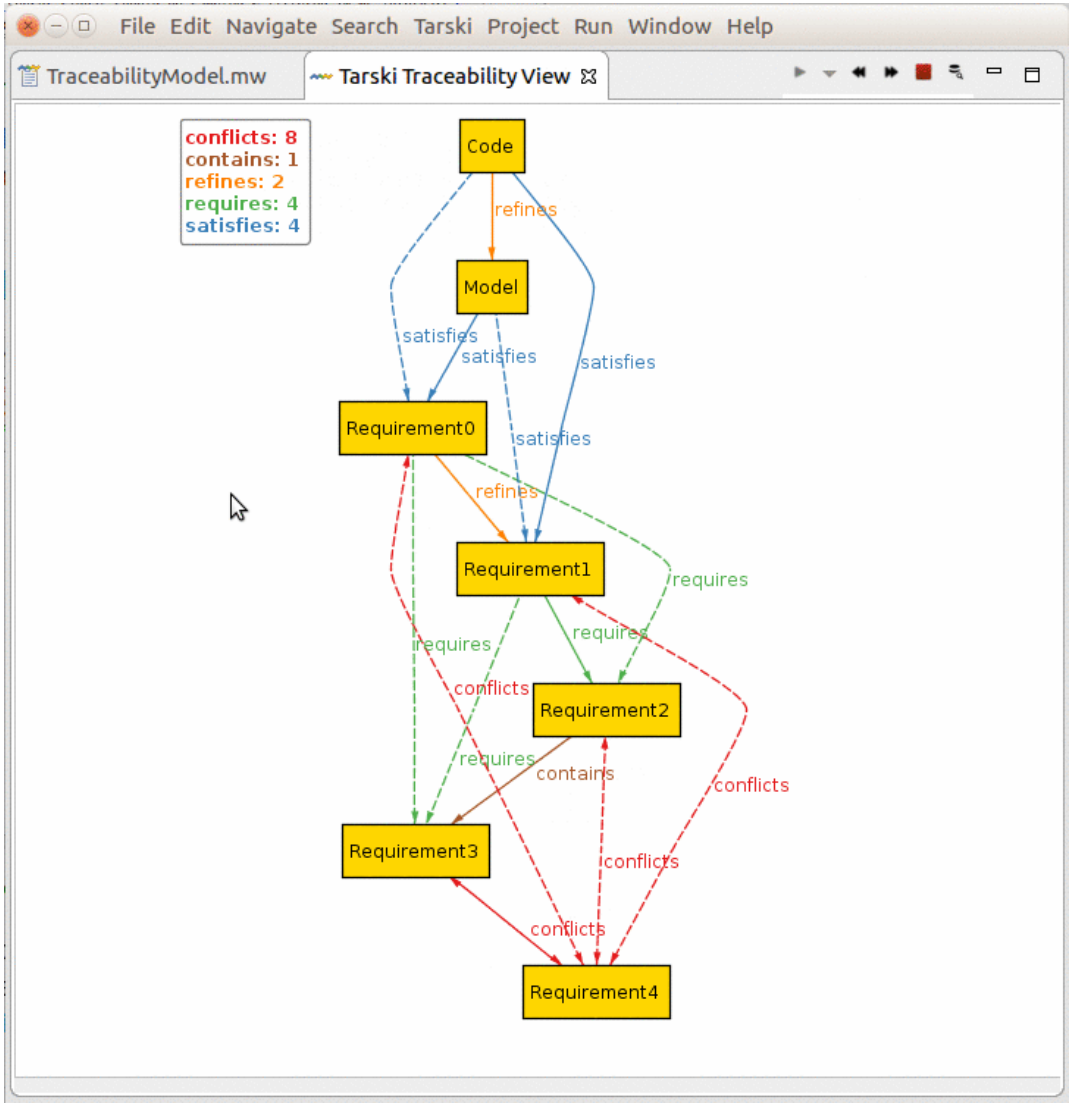
```
1  # Process Height Sensor Signal/Voltage
2
3  The height sensor signal represents the mechanical suspension/vehicle height
   position and is one of the most critical signals of the system. Based on the
   value of the signal, the system should make height corrections by raising or
   lowering vehicle until ride height is met.
4
5  ## REQ_60
6
7  R60: The system  shall do height corrections using long and short term filtered
   height sensor signal with 10ms interval.
8
9  ## REQ_11
10
11 R11: The system shall do height corrections using long and short term filtered
   height sensor signal.
12
13 ## REQ_97
14
15 R97: The system shall filter height sensor signal in short term and long term for
   height corrections.
16
17
18 ## REQ_98
19
20 R98: The system shall filter height sensor signal in long term for height
   corrections.
21
22
23 ## REQ_59
24
25 R59: The system shall always use height sensors in the range of 0-5V to avoid
   long term signal filtering.
26
```

Markdown Source | Preview

② Assigning Traces between the Project Artifacts



③ Reason about Traces in the Artifacts



Comparisons of Several Use Cases for Trace Inferring

	Trace Types	Facts	Traced Elements	Manual Traces	Inferred Traces	Inconsis. Parts
#1	7	11	125	138	502	3
#2	11	20	47	102	145	5
#3	10	14	16	21	53	1

	Artifacts	Traces	Inferred	Alloy	KodKod	Z3
#1	123	102	89	67922	25668	40900
#2	56	27	25	4428	84	480
#3	42	103	75	724	1	1460

Conclusion and Future Work

- Should we consider also the temporal behavior of the traceability? Interesting analysis scenarios exist in industry
- We are not supporting ordered sets of Alloy which usually help model the dynamic behaviour.
- First-order theory of relations might be a candidate for modeling traceability in Multi-pardigm Modeling settings. However, DPLL(T) solvers does not currently exists for this fragment of the theory.
- Alloy Language is too expressive for the domain of traceability. We're working on the formalization of a First-order theory for traceability and the development of a domain-specific language for traceability.

For Further Reading

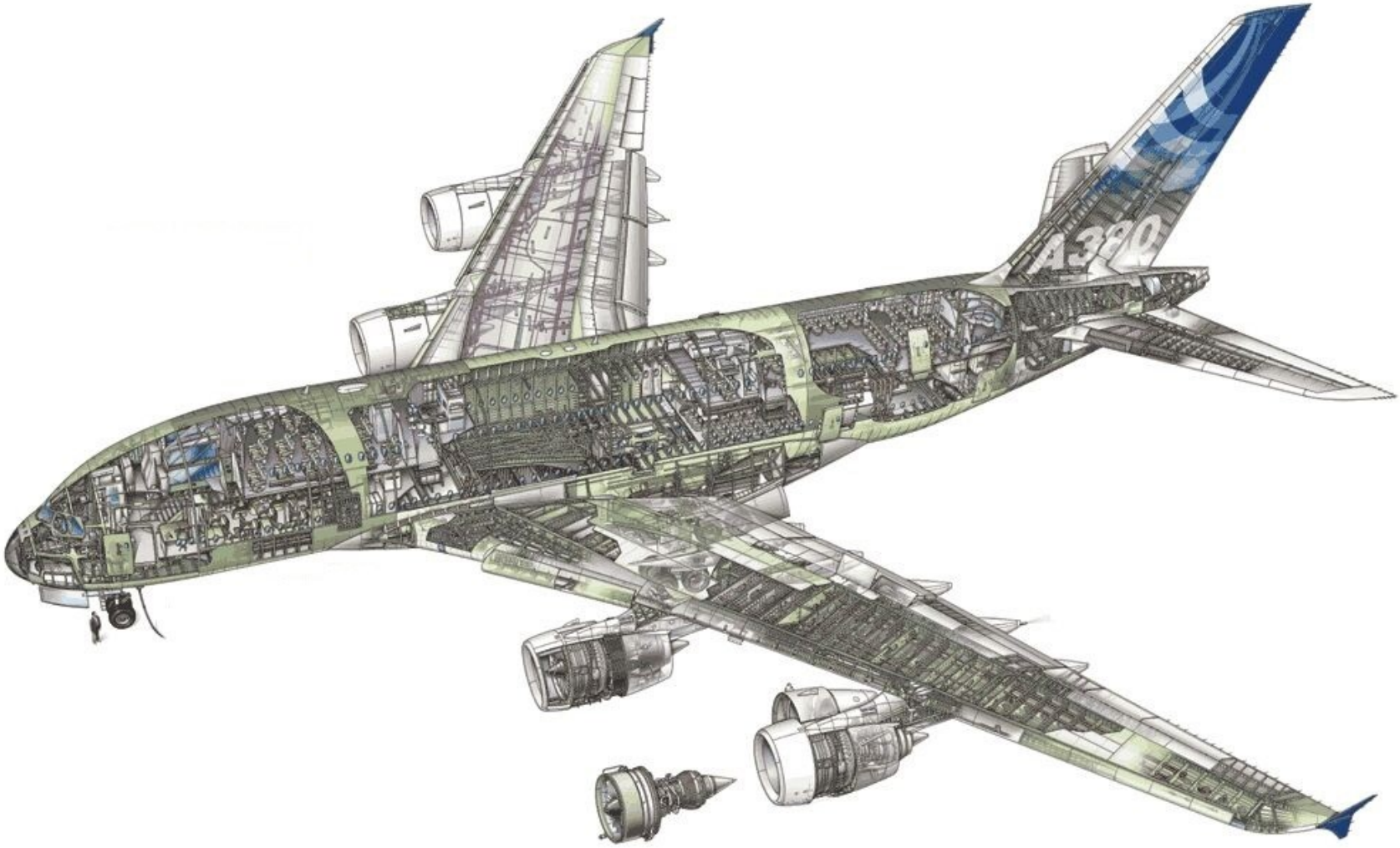
F. Erata et. al.
ModelWriter: Text and model-synchronized document engineering platform
32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17), pp. 928-933, 2017

F. Erata et. al.
Tarski: A platform for automated analysis of dynamically configurable traceability semantics
The 32nd ACM SIGAPP Symposium On Applied Computing (SAC'17), pp. 1607-1614, 2017

Source codes, datasets and screencasts
`https://modelwriter.github.io/Tarski/`

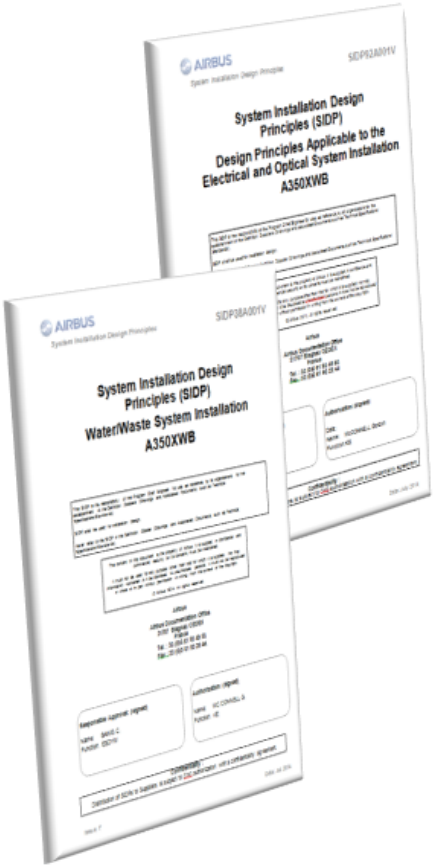
Airbus Group Innovations

System Installation Design Principles



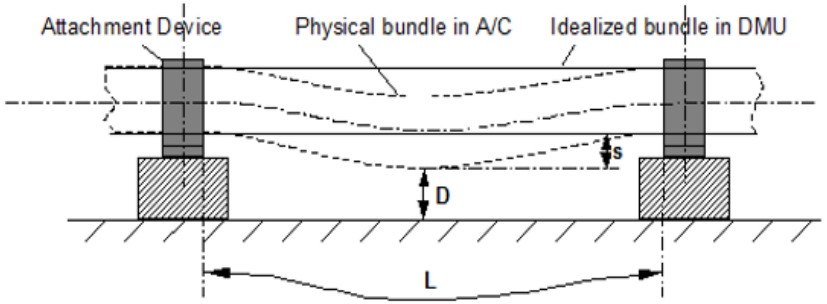
Airbus Group Innovations

System Installation Design Principles



SIDP92A001V-A-784

For installation of optical and electrical harnesses additional clearance for sagging (s) shall be provided as detailed below:



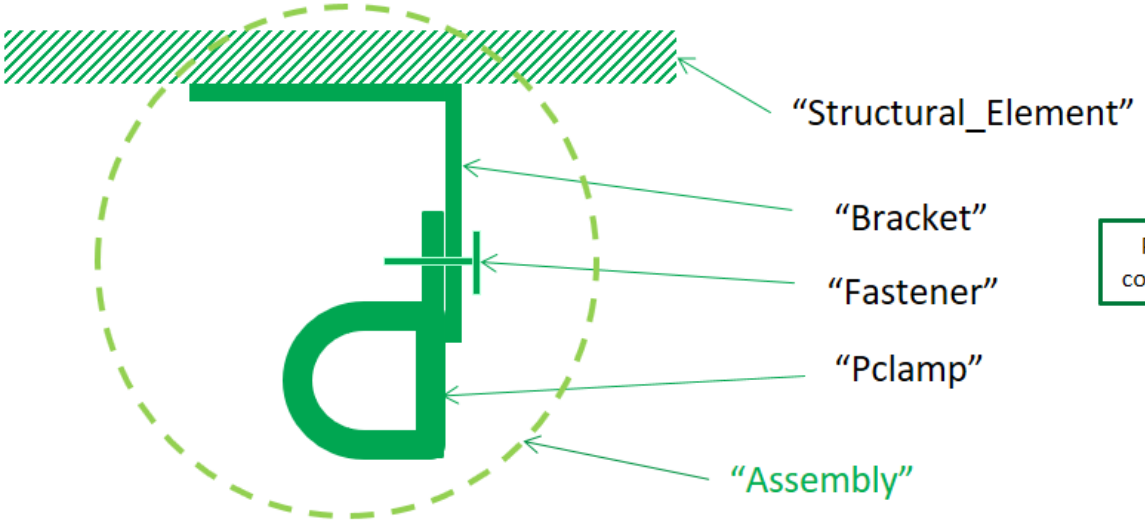
s...Sagging of bundle (real behavior of physical bundle in A/C due to gravity, ageing, etc.)
D...Required Distance
L...Actual length of a bundle segment between two Attachment Points (as designed in DMU)

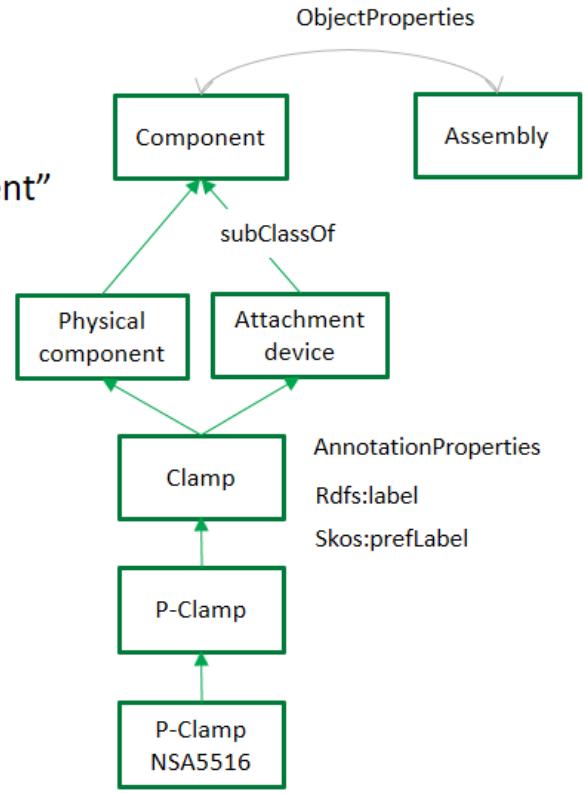
Figure 6: Sagging of bundles between attachment points

Note: Unless the bundle has a straight routing, L is bigger than the pitch between the Attachment Points.

Airbus Group Innovations

System Installation Design Principles





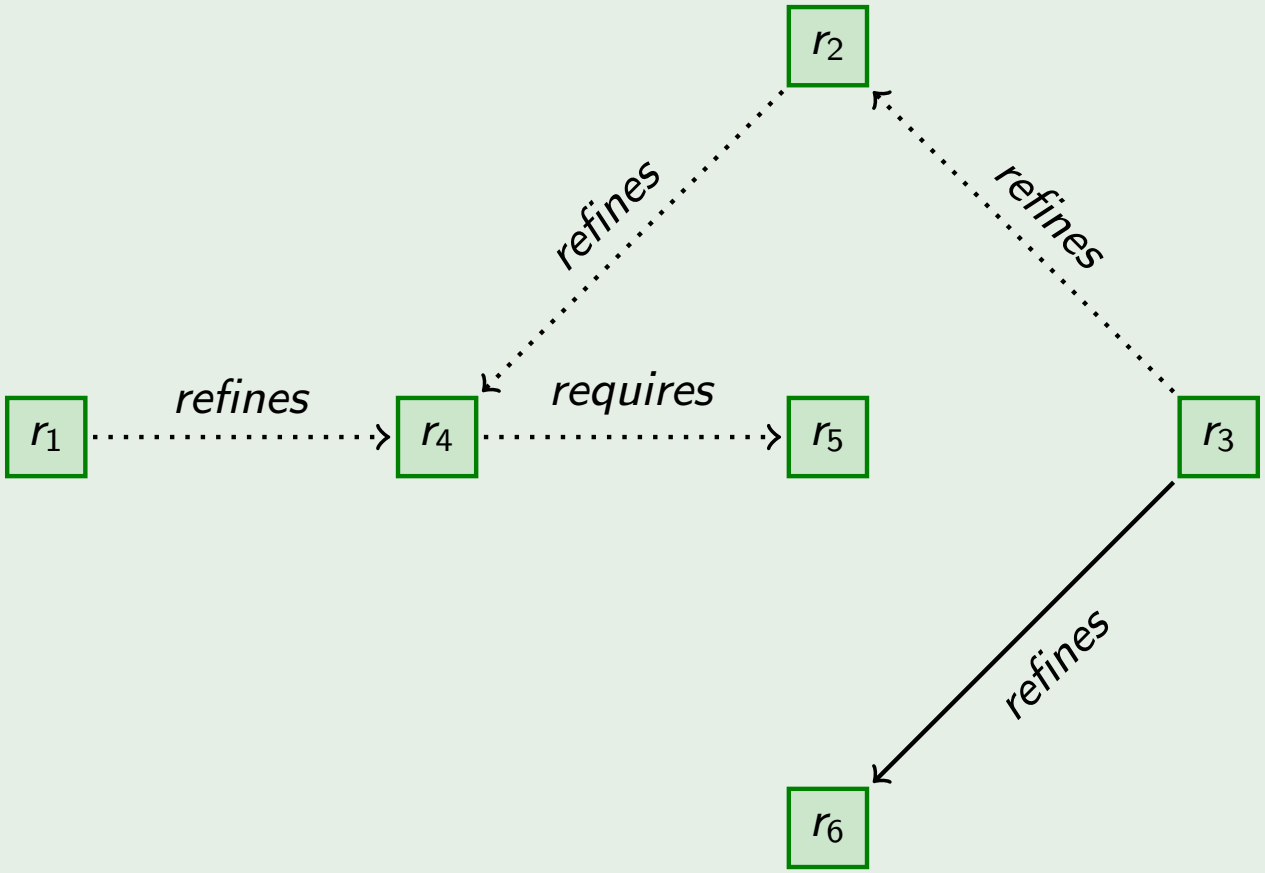
"P-clamp NSA5516 can be fixed on X with Y"

"Physical component" "Standard reference"

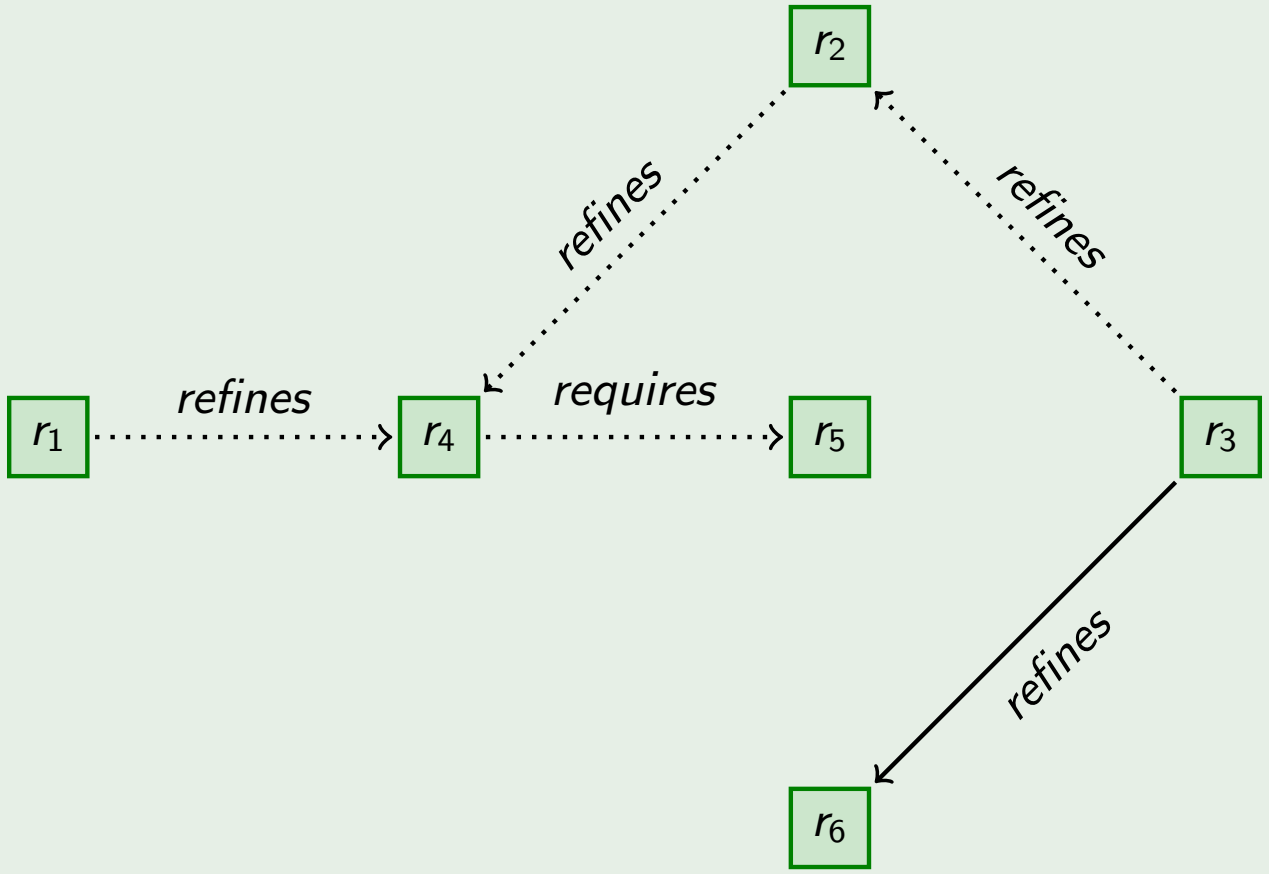
System Installation Design Principles

r_1	*Bracket* shall be used in *hydraulic area Alpha*
r_2	*Adhesive bonded bracket* shall be used in *hydraulic area*
r_3	*Adhesive bonded bracket* shall be used in *hydraulic area Alpha*
r_4	*Bracket* shall be used in *hydraulic area*
r_5	*Bracket* shall be installed in *hydraulic area*
r_6	*Bracket* shall be installed in *fuel tank*

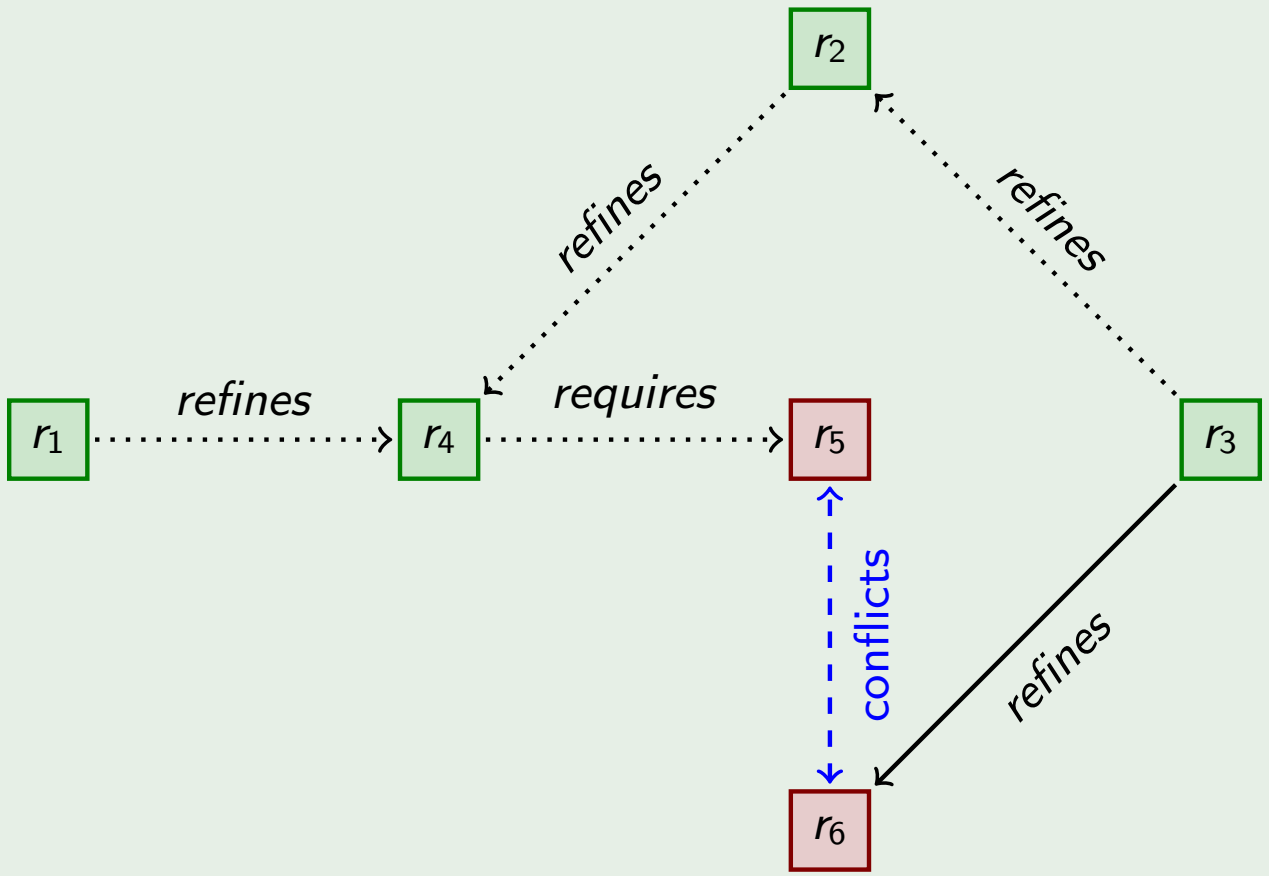
Illustration of "conflict" propagation operationally in the system



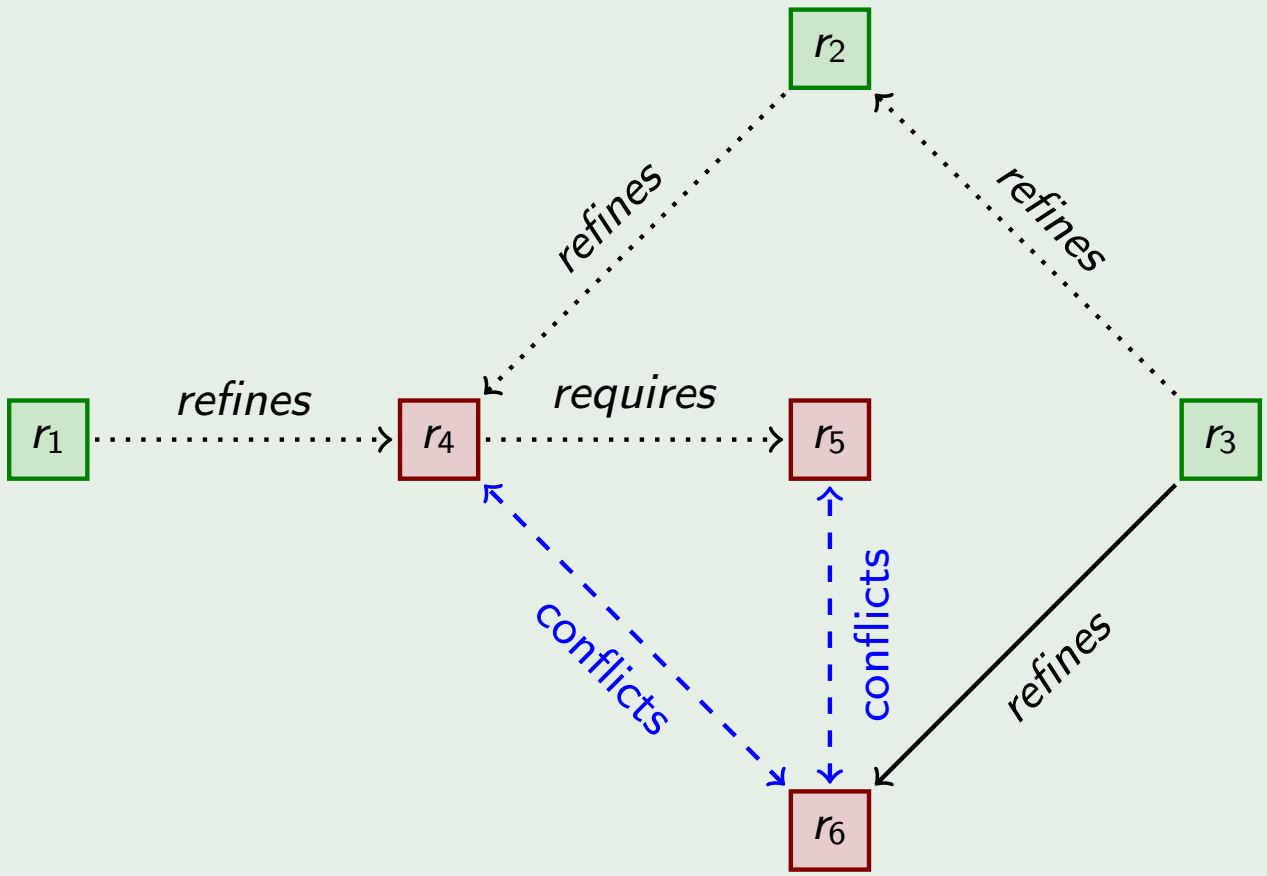
Current Traceability View



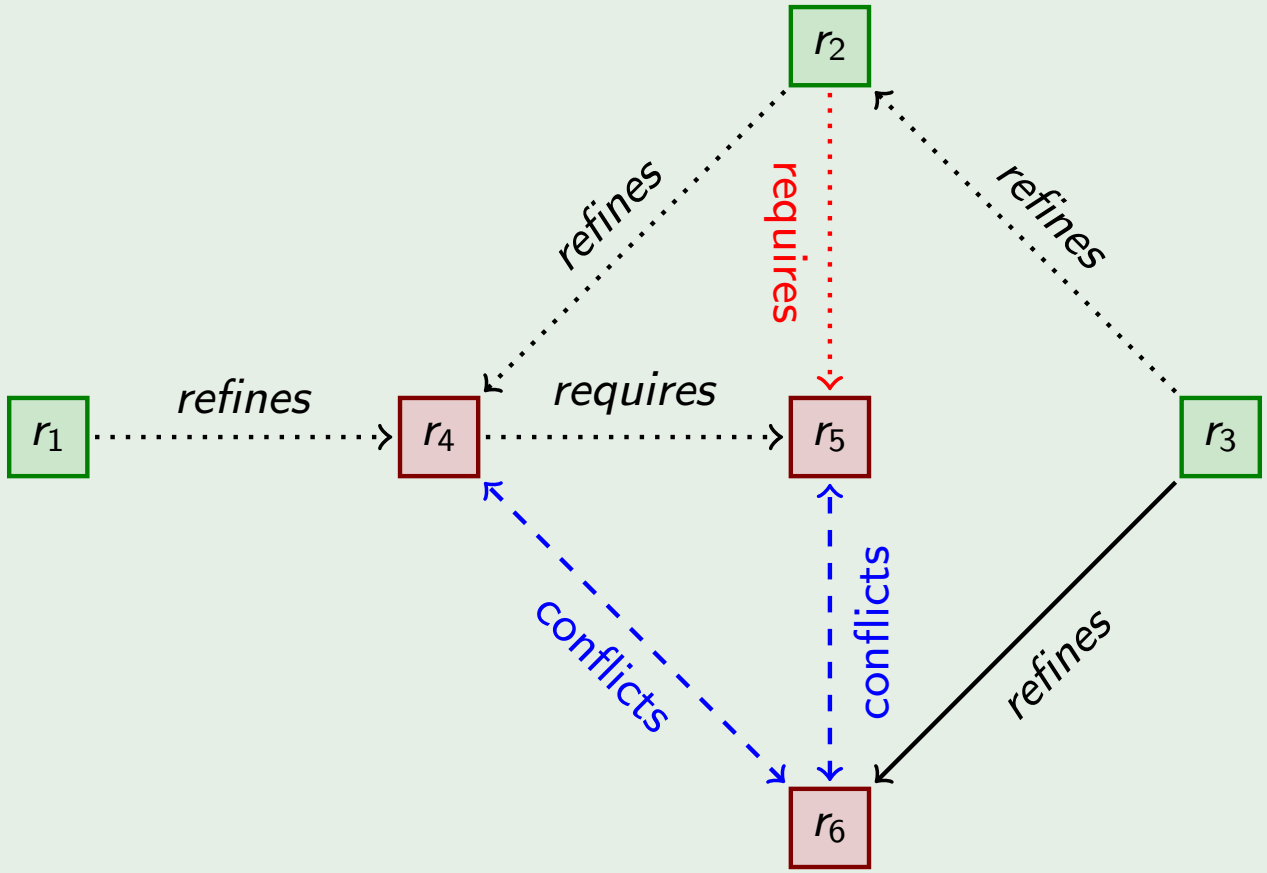
Semantic Parser identifies a conflict between r_5 and r_6



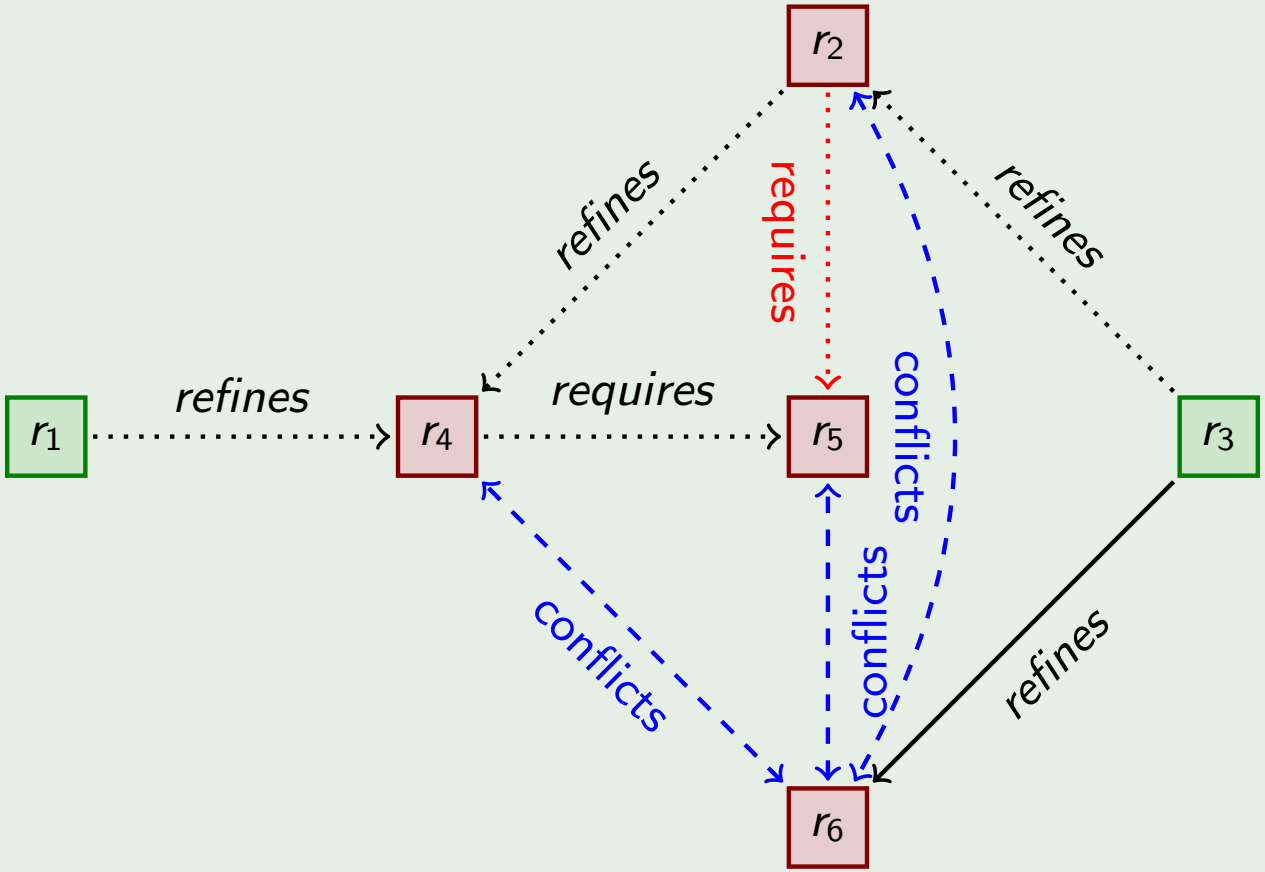
Tarski infers a conflict between r_6 and r_4



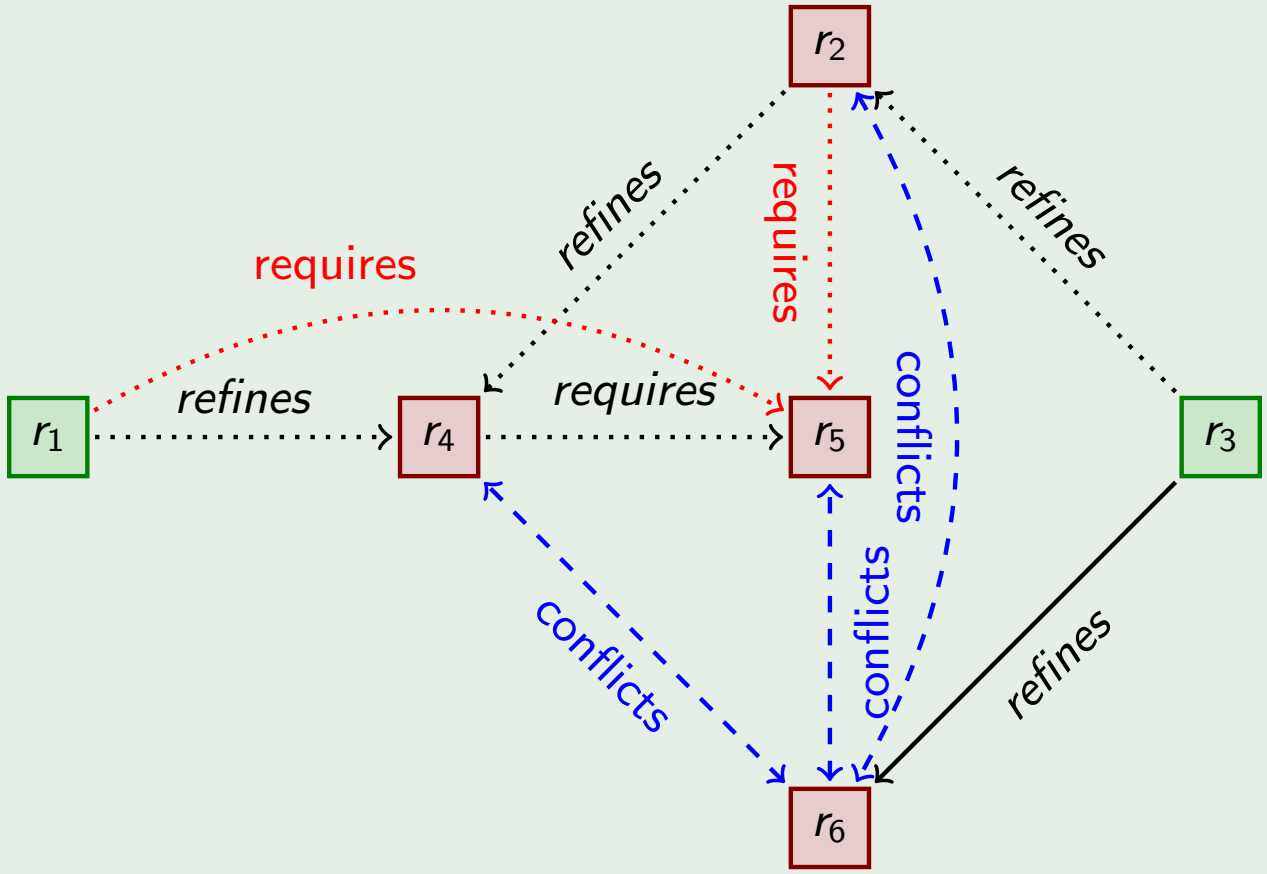
Tarski infers a require relation from r_2 to r_5



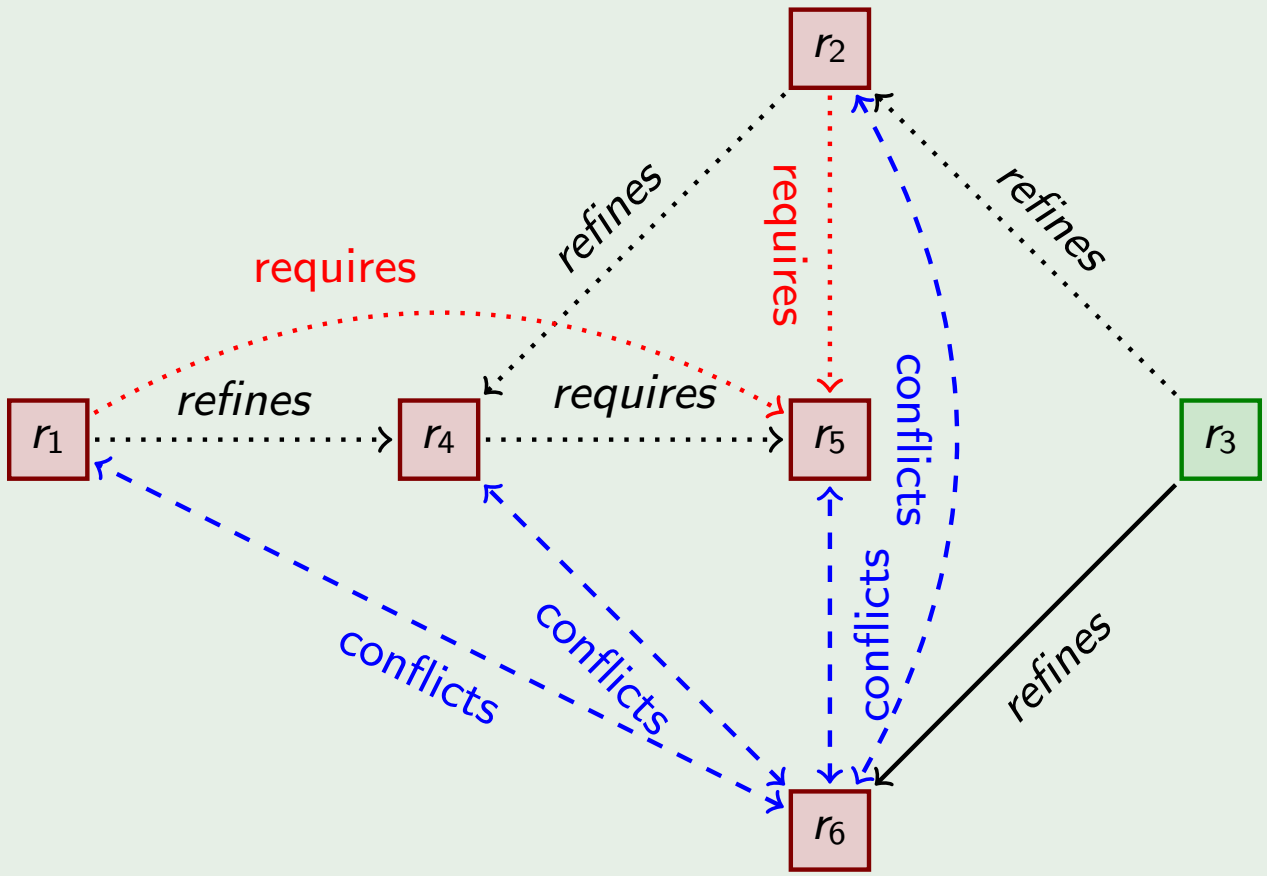
Tarski infers a conflict between r_2 and r_5



Tarski infers a require relation from r_1 to r_5



Tarski infers a conflict between r_1 and r_6



Axiomatization of the Traceability Theory

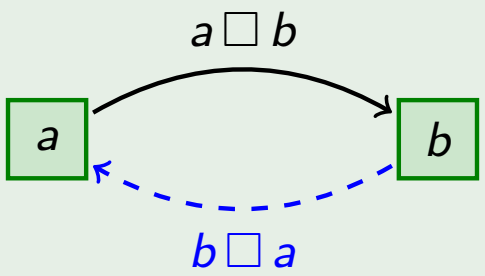
$TH(Traceability)$

$$\Sigma_T : \{=, \in\} \cup \Sigma_T^1 \cup \Sigma_T^2$$
$$\Sigma_T^1 : \{Artifact, Requirement, Specification\}$$
$$\Sigma_T^2 : \{requires, refines, contains, equals, conflicts\}$$

Symmetric Relations

$$\vdash \forall a, b \in A \mid (a, b) \in \square \rightarrow (b, a) \in \square$$
$$\square = \mathit{conflicts} \cup \mathit{equals}$$

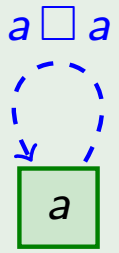
Model



Reflexive relations

$$\vdash \forall a \in A \mid (a, a) \in \square$$
$$\square = \text{equals}$$

Model

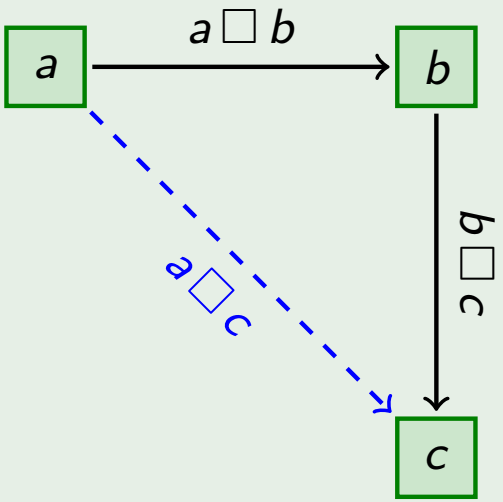


Transitive relations

$$\vdash \forall a, b, c \in A \mid (a, b) \in \square \wedge (b, c) \in \square \rightarrow (a, c) \in \square$$

$$\square := \text{contains} \oplus \text{requires} \oplus \text{refines} \oplus \text{equals} \oplus \text{p-refines}$$

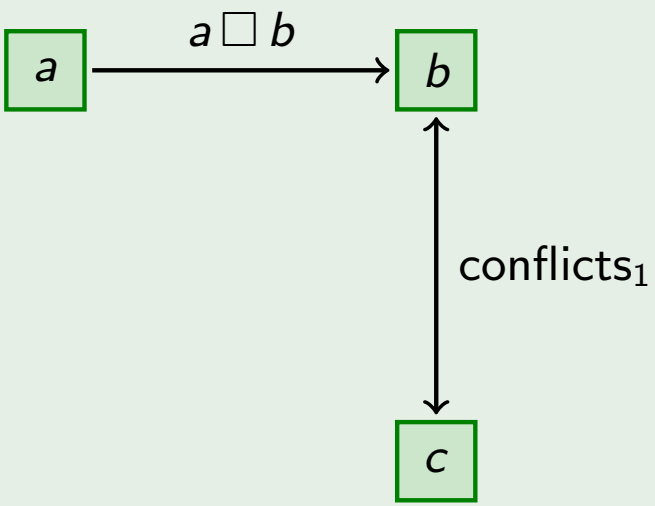
Model



Inferring "conflicts" relation

$$\vdash \forall a, b, c \in A \mid [(a, b) \in \square \wedge (b, c) \in \text{conflicts}] \rightarrow (a, c) \in \text{conflicts}$$
$$\square := \text{requires} \oplus \text{refines} \oplus \text{contains}$$

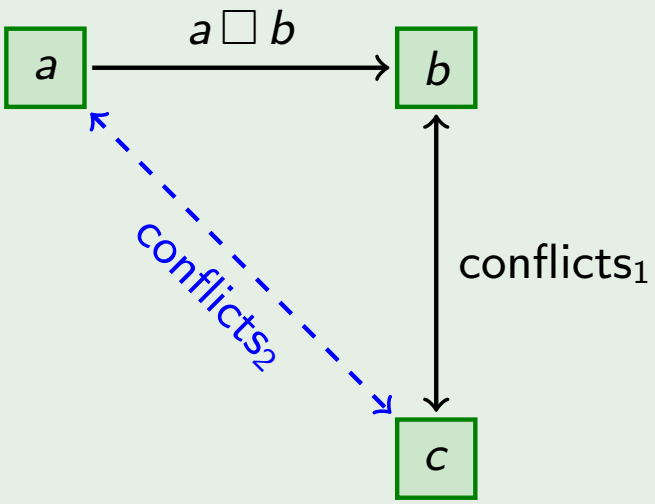
Partial Model



Inferring "conflicts" relation

$$\vdash \forall a, b, c \in A \mid [(a, b) \in \square \wedge (b, c) \in \text{conflicts}] \rightarrow (a, c) \in \text{conflicts}$$
$$\square := \text{requires} \oplus \text{refines} \oplus \text{contains}$$

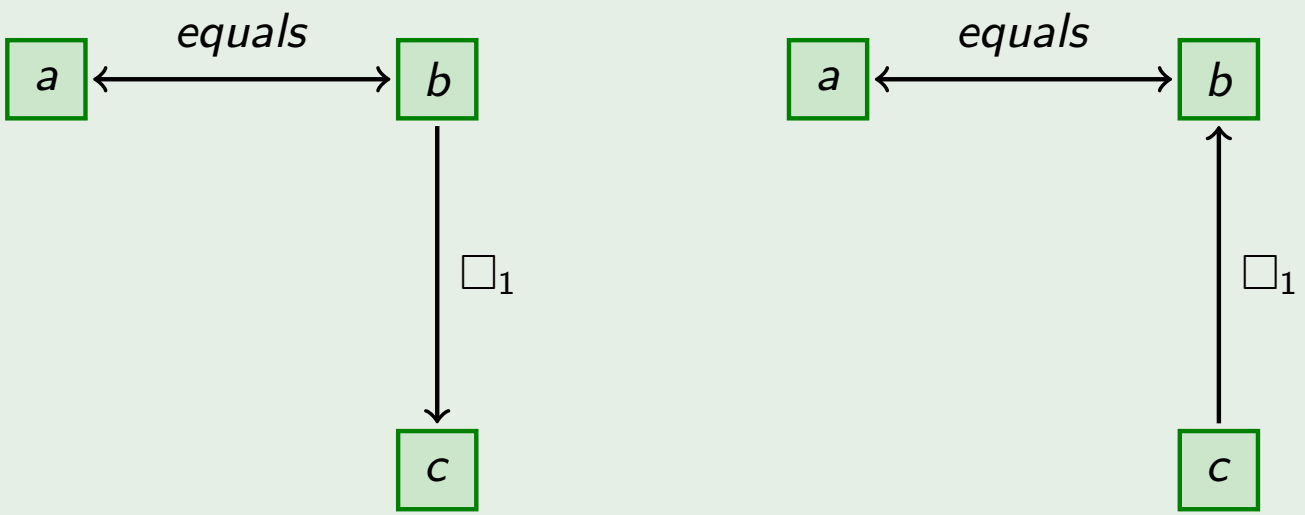
Complete Model



Reasoning about equality

$$\vdash \forall a, b, c \in A \mid (a, b) \in \text{equals} \wedge (b, c) \in \square \rightarrow (a, c) \in \square$$
$$\vdash \forall a, b, c \in A \mid (a, b) \in \text{equals} \wedge (c, b) \in \square \rightarrow (c, a) \in \square$$

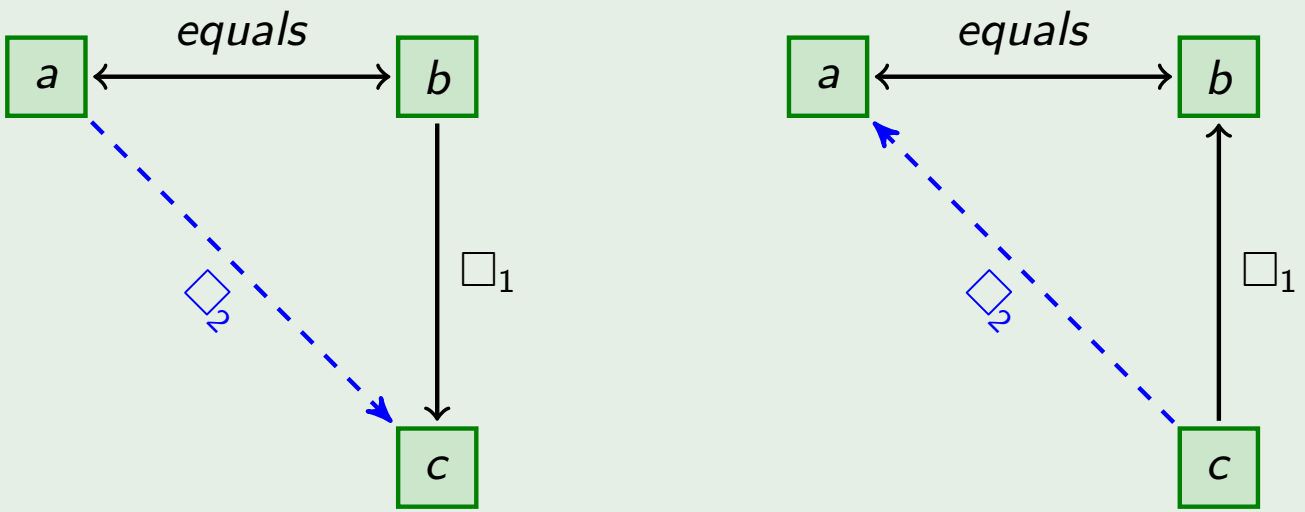
Partial Model



Reasoning about equality

$$\vdash \forall a, b, c \in A \mid (a, b) \in \text{equals} \wedge (b, c) \in \Box \rightarrow (a, c) \in \Box$$
$$\vdash \forall a, b, c \in A \mid (a, b) \in \text{equals} \wedge (c, b) \in \Box \rightarrow (c, a) \in \Box$$

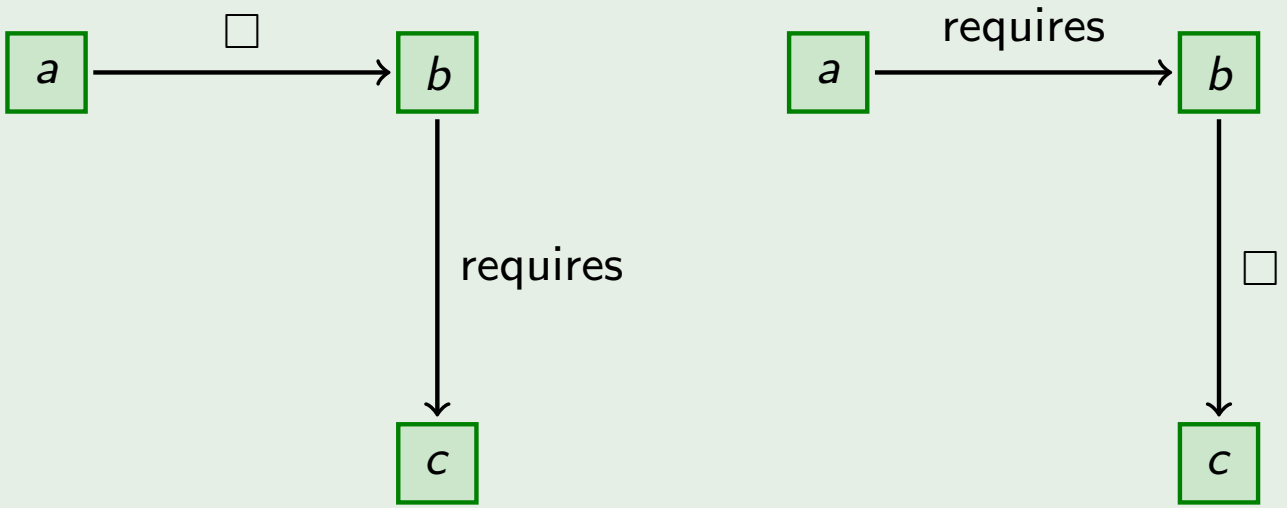
Complete Model



Inferring "requires" relation with "contains" and "refines"

$$\vdash \forall a, b, c \in A \mid (a, b) \in \square \wedge (b, c) \in \text{requires} \rightarrow (a, c) \in \text{requires}$$
$$\vdash \forall a, b, c \in A \mid (a, b) \in \text{requires} \wedge (b, c) \in \square \rightarrow (a, c) \in \text{requires}$$
$$\square := \text{refines} \oplus \text{contains}$$

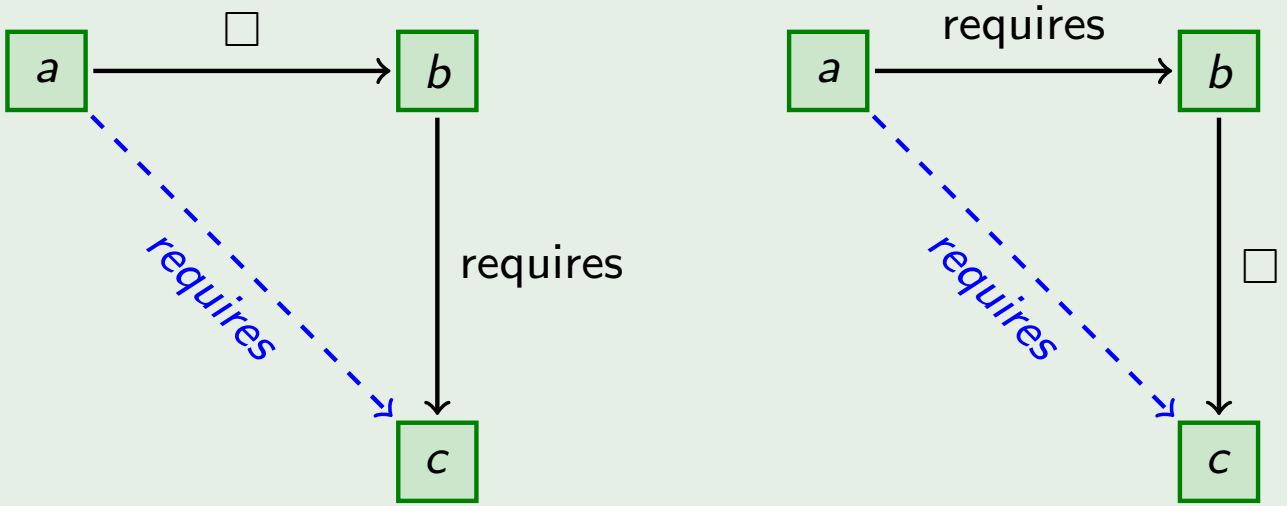
Partial Model



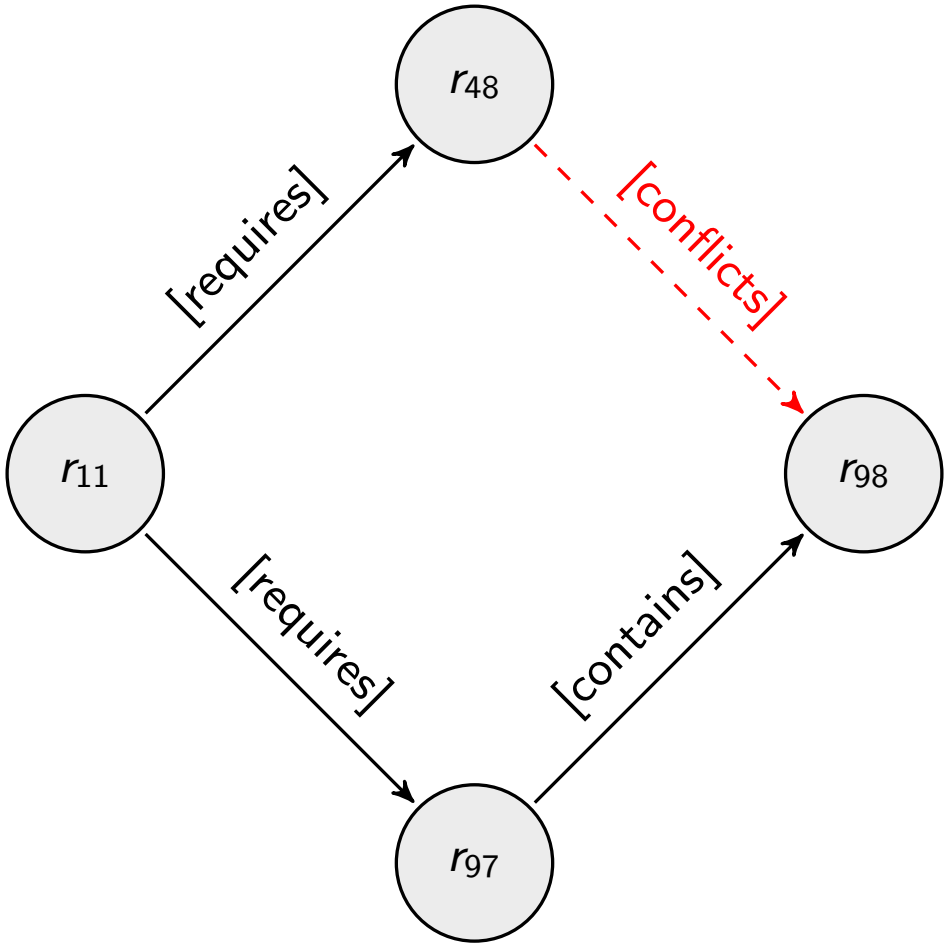
Inferring "requires" relation with "contains" and "refines"

$$\vdash \forall a, b, c \in A \mid (a, b) \in \square \wedge (b, c) \in \text{requires} \rightarrow (a, c) \in \text{requires}$$
$$\vdash \forall a, b, c \in A \mid (a, b) \in \text{requires} \wedge (b, c) \in \square \rightarrow (a, c) \in \text{requires}$$
$$\square := \text{refines} \oplus \text{contains}$$

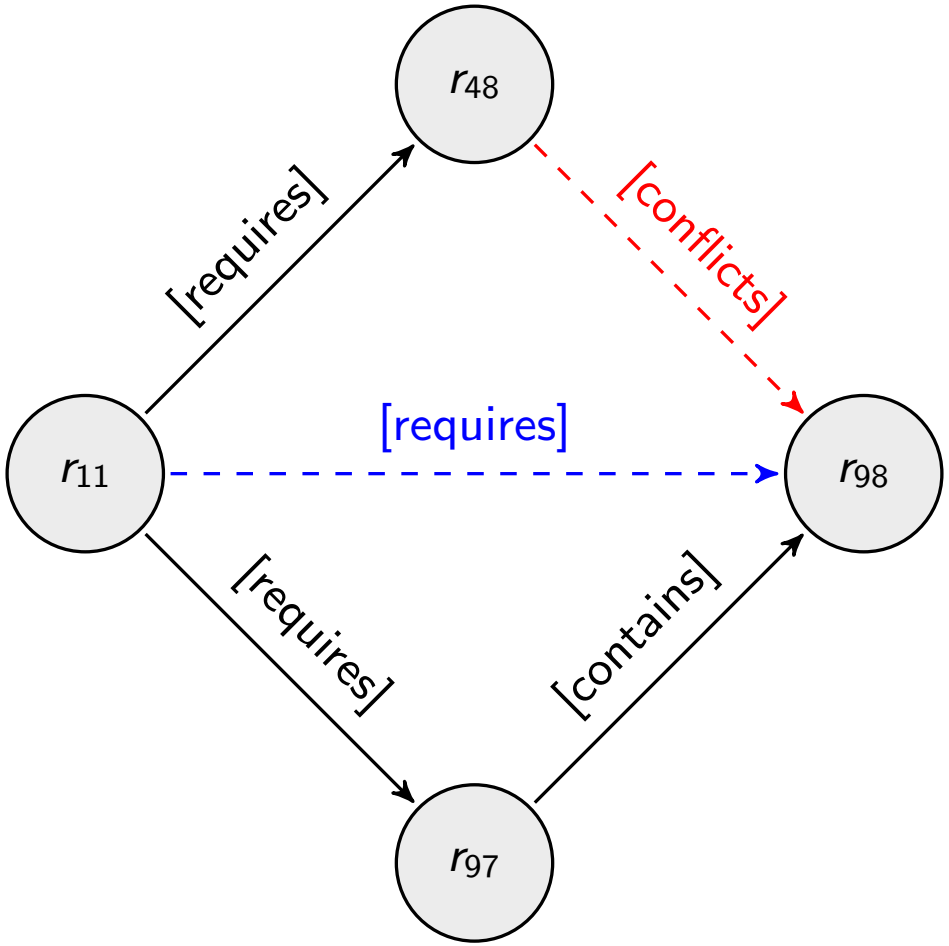
Complete Model



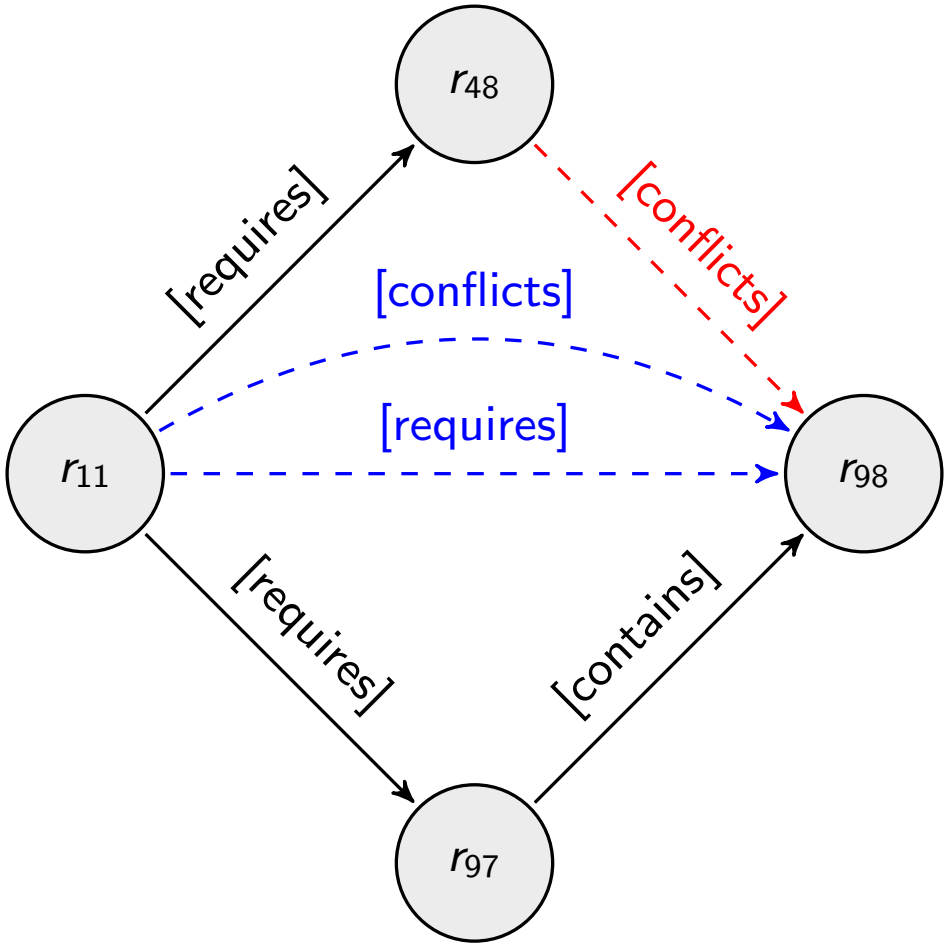
ECAS - Unsatisfiable Core 1



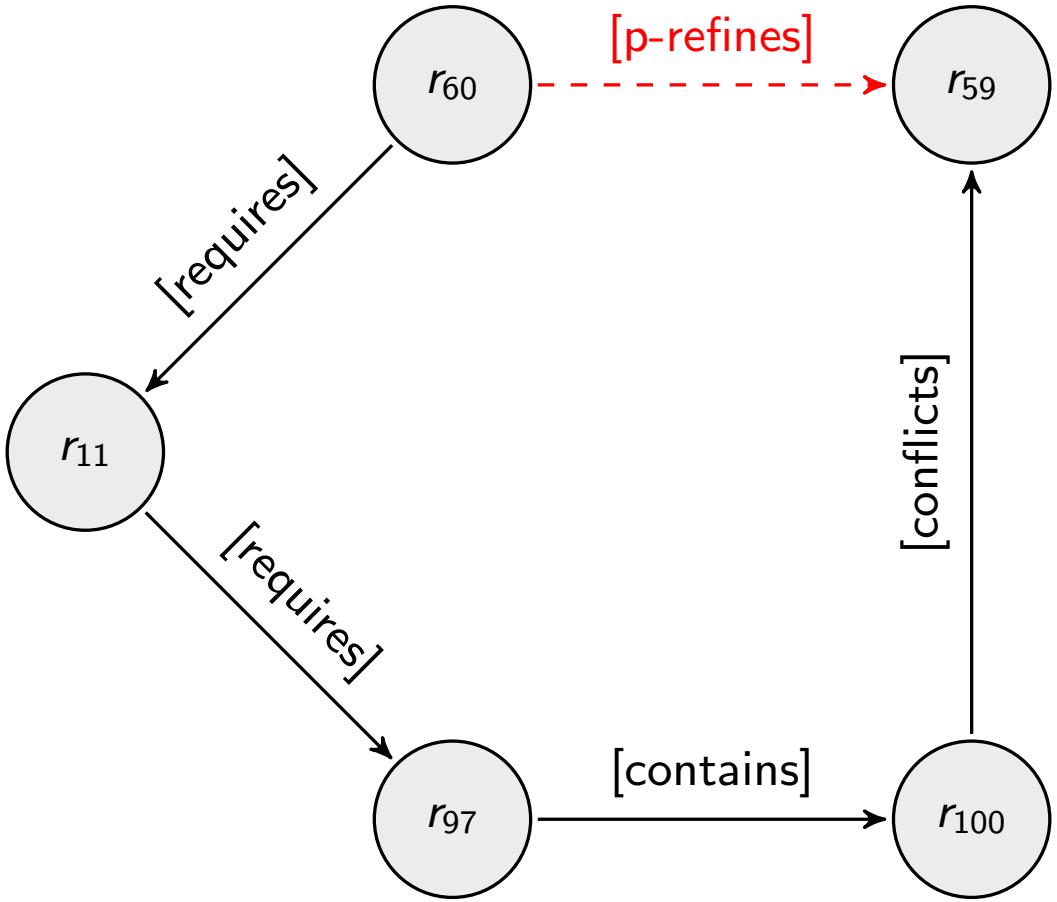
ECAS - Unsatisfiable Core 1



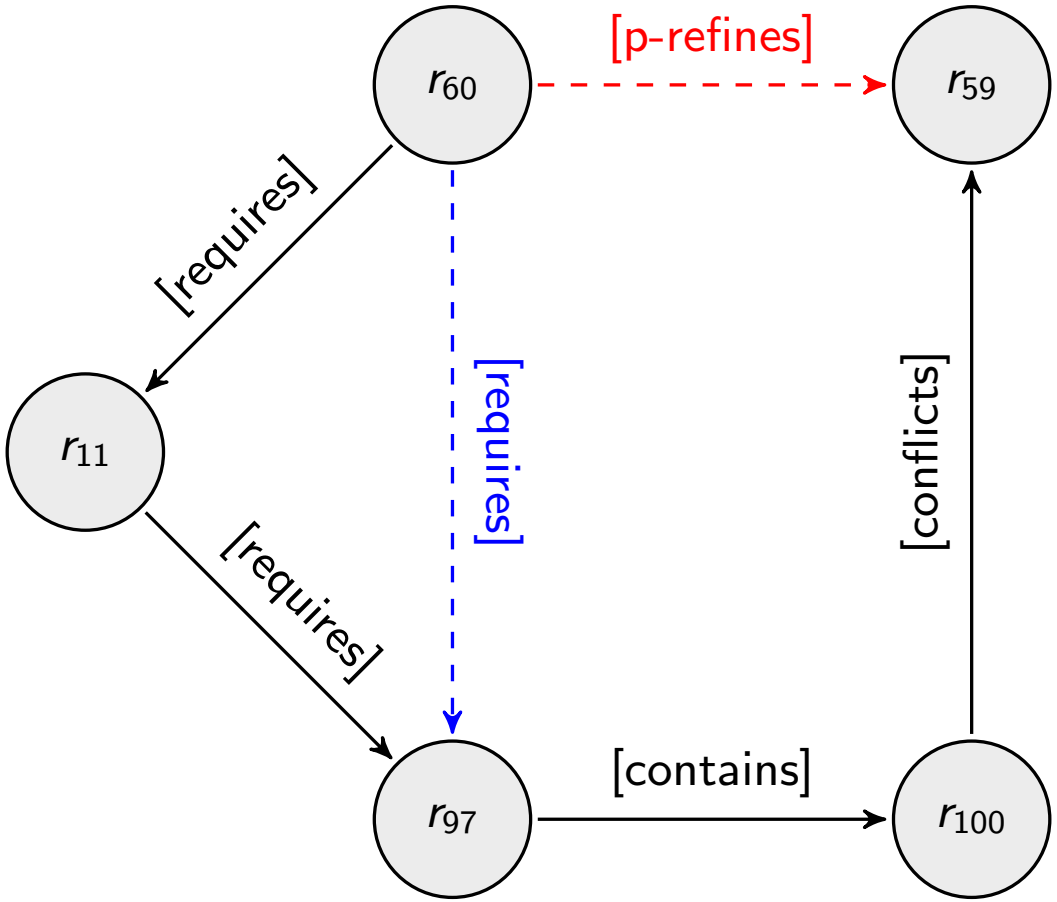
ECAS - Unsatisfiable Core 1



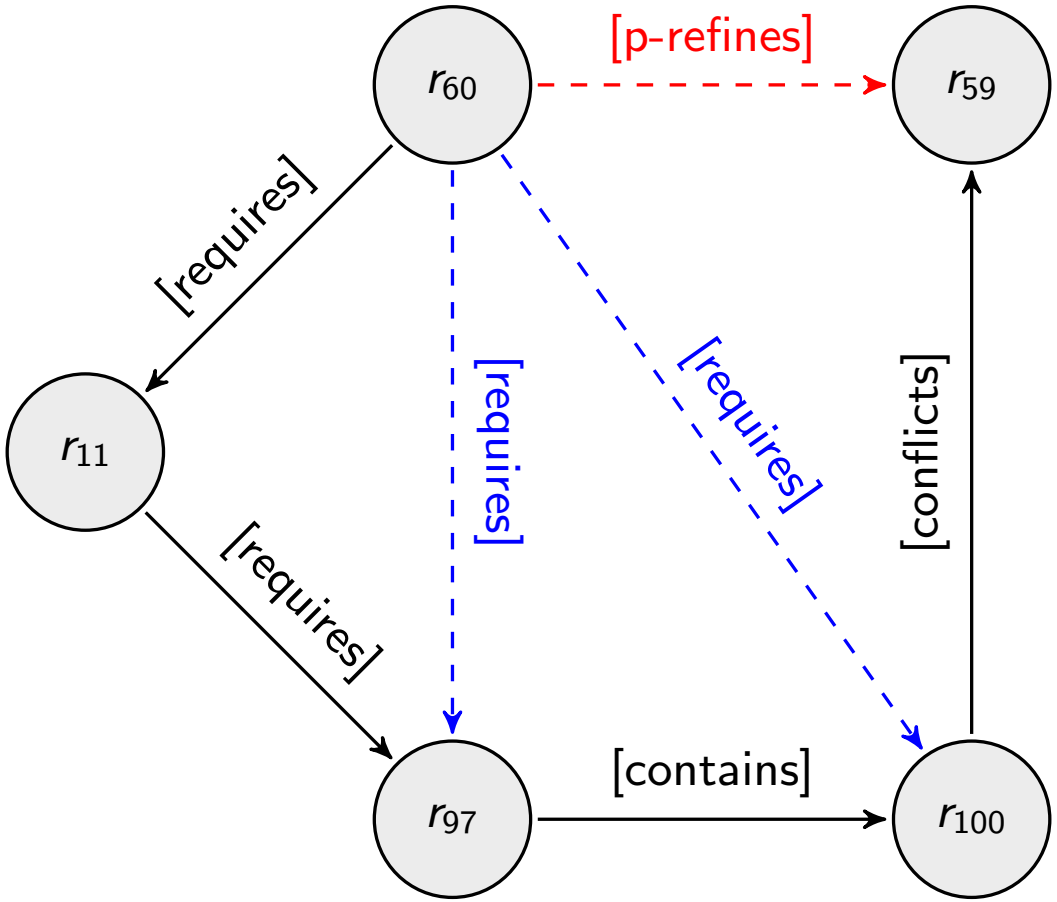
ECAS - Unsatisfiable Core 2



ECAS - Unsatisfiable Core 2



ECAS - Unsatisfiable Core 2



ECAS - Unsatisfiable Core 2