

Participatory Verification of Railway Infrastructure by Representing Regulations in RailCNL

Bjørnar Luteberget / John J. Camilleri /
Christian Johansen / Gerardo Schneider

SEFM 2017, Sep 6-8, 2017



UiO • **University of Oslo**



RailCOMPLETE



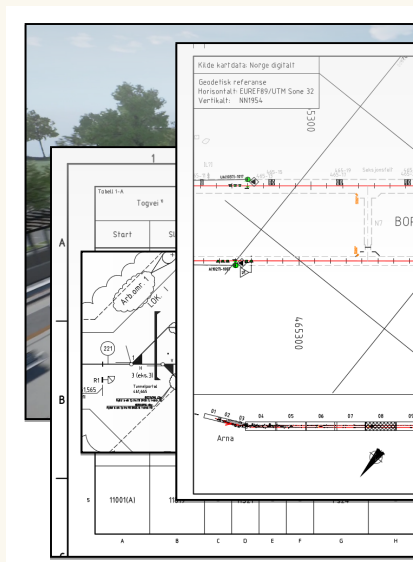
CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Background: railway engineering

- ▶ **Costly** projects with high quality requirements, **complicated** regulations.
- ▶ Produce a lot of tables, drawings, 3D models, specifications, documentation, etc.
- ▶ Evaluation relies on a lot of manual checking of regulations **compliance**.
- ▶ Coordination between **disciplines** require constant **re-evaluation** of designs.



RailCons project: automated verification

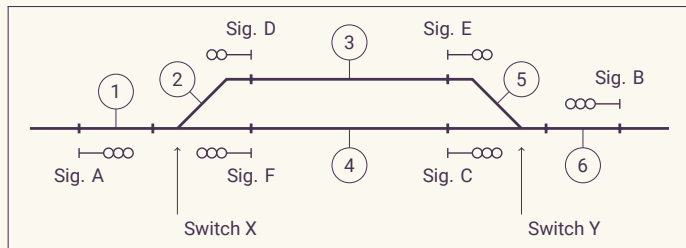
Project objectives:

- ▶ Verify that railway signalling and interlocking **designs** **comply with regulations**.
- ▶ Provide tools which allow **railway engineers** to perform such verification as part of their **daily** routine (“lightweight verification”).

“Formal methods will never have a significant impact until they can be used by people that **don't understand** them.”

– (attributed to) Tom Melham

Models: railway signalling and interlocking designs



(a) Track and signalling component layout


Route	Start	End	Sw. pos	Detection sections	Conflicts
AC	A	C	X right	1, 2, 4	AE, BF
AE	A	E	X left	1, 2, 3	AC, BD
BF	B	F	Y left	4, 5, 6	AC, BD
BD	B	D	Y right	3, 5, 6	AE, BF

(b) Tabular interlocking specification

Properties: technical regulations

- ▶ In our case study: Norwegian regulations from national railways (Bane NOR)
- ▶ **Static** kind of properties, often related to object properties, topology and geometry (example on next slide)

e) Dersom nødvendig stopplengde er lengre enn avstanden mellom to etterfølgende hovedsignal, skal det benyttes gjennomsignalering ved hjelp av ATC (Signal/Prosjektering/ATC), se Figur 7 [↗](#).



Figur 7: Forsignalsplassering og gjennomsignalering [↗](#)

f) Et forsignal skal plasseres på foregående hovedsignals mast dersom avstanden mellom det tilhørende hovedsignalet og det foregående hovedsignalet er ≤ 2200 meter.

g) Mellom et forsignal og det tilhørende hovedsignalet skal det ikke plasseres andre hoved- eller forsignal.

h) Et forsignal skal plasseres slik at siktavstanden oppfyller kravene til enten "brutt sikt" eller til "ubrutt sikt" i Tabell 4 [↗](#).

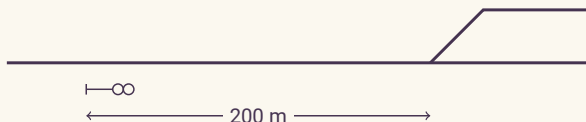
Tabell 4: Siktkrav til forsignal

Sikt	Strekningens høyeste tillatte kjørehastighet [km/h]																		
	40	45	50	55	60	65	70	75	80	85	90	95	100	105	110	115	120	125	≥ 130
Ubrutt	78	88	97	107	117	126	136	146	156	165	175	185	194	204	214	224	233	243	250

Properties: technical regulations

Example from regulations:

- ▶ A *home main signal* shall be placed at least **200 m** in front of the first controlled, **facing switch** in the entry train path.

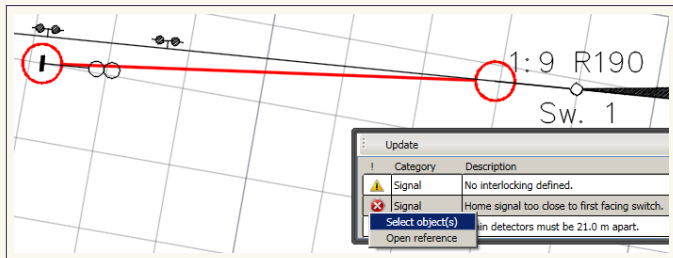


- ▶ Can be classified as follows:
 - Object properties
 - Topological layout properties
 - Geometrical layout properties
 - Interlocking properties

Datalog verification tool

- ▶ Prototype using **XSB Prolog** tabled predicates, front-end is the **RailCOMPLETE** tool based on Autodesk AutoCAD
- ▶ Rule base in Prolog syntax with **structured comments** giving information about rules

```
%| rule: Home signal too close to first facing switch.  
%| type: technical  
%| severity: error  
homeSignalBeforeFacingSwitchError(S, SW) :-  
    firstFacingSwitch(B, SW, DIR),  
    homeSignalBetween(S, B, SW),  
    distance(S, SW, DIR, L), L < 200.
```



Challenge: participatory verification

Challenge: Users (railway engineers) are not experts in verification techniques, so how can they

- ▶ **build** models of the systems to be verified?
- ▶ **write** properties in the verifier's input language?
- ▶ **interpret** the output of the verifier when violated properties are found?

Input to verification:

- ▶ **Models**: CAD extended with structured railway data (familiar to engineers, user-friendly)
- ▶ **Properties**: **Datalog** (unfamiliar to engineers, not user-friendly enough)

... consider another **verification property input language**?

REMU project – Chalmers/GU Gothenburg

REMU project: Reliable Multilingual Digital Communication –

- ▶ Goals (among others): **grammar development**, testing, analysis.
- ▶ Tools: **Grammatical Framework** – Programming language for multilingual **grammar applications**.
- ▶ **Controlled natural language**
Controlled natural languages (CNLs) are subsets of natural languages that are obtained by restricting the grammar and vocabulary in order to reduce or eliminate ambiguity and complexity.

Grammatical Framework

Define domain model in an **abstract** syntax, define one or more mappings to text in a **concrete** syntax.

Abstract syntax:

- ▶ Domain-specific tree data structure for representing the desired content.

```
abstract ToyRailway = {  
  cat Subject; Length; Restriction; Statement;  
  fun Signal, Switch, Detector : Subject;  
    LengthMeters : Int -> Length;  
    GreaterThan, LessThan : Length -> Restriction;  
    ObjectSpacing : Subject -> Subject -> Restriction  
      -> Statement; }
```

- ▶ Example phrase in abstract syntax:

```
ObjectSpacing Signal Switch (GreaterThan (LengthMeters 20))
```

Grammatical Framework

Concrete syntax:

- ▶ A mapping from the abstract syntax to text.
- ▶ Invertible, so a GF concrete syntax gives you a parser and a linearization (generator).

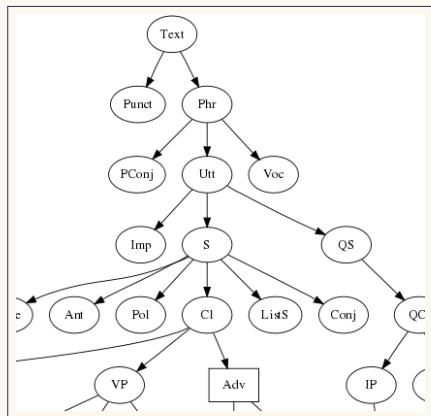
```
concrete ToyRailwayEng of ToyRailway = {  
  lincat Subject = Str; Length = Str; (...)  
  lin Signal = "signal"; (...)  
    LengthMeters i = i ++ "m"  
    GreaterThan l = "more than" ++ l  
    ObjectSpacing o1 o2 r =  
      "a" ++ o1 ++ "must be" ++ r  
      ++ "from a" ++ o2; }
```

- ▶ Parse: "a signal must be more than 20 m from a switch"
ObjectSpacing Signal Switch (GreaterThan (LengthMeters 20))
- ▶ Complexity and constraints of natural language quickly becomes infeasible to handle when the language grows...

Grammatical Framework's Resource Grammars

Comprehensive linguistic model of natural languages with a unified API for forming sentences.

- ▶ Parse/generate in 31 languages using a unified API.
- ▶ Ensures grammatical correctness of phrases using the type system.



API usage example:

```
OrientationAngleTo vec =  
    mkCN (mkCN angle_N  
          (mkAdv to_Prep (mkNP the_Det vec)));
```

Related work

Domain-specific languages for railway verification:

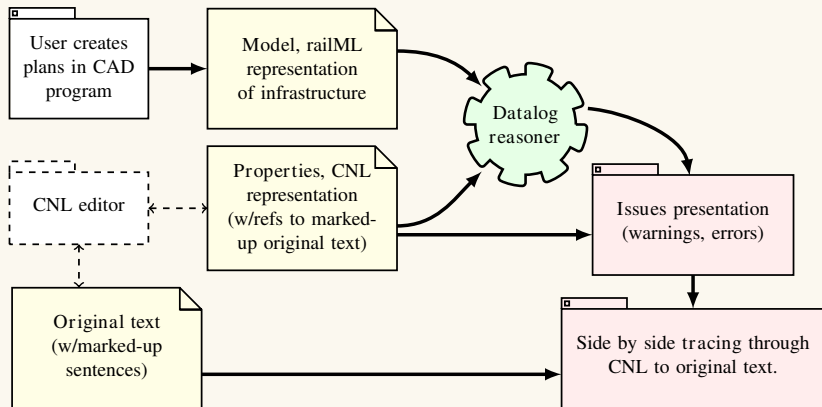
- ▶ Verification of implementation of railway **control systems** (Vu, Haxthausen, Peleska, 2014). Concise verification properties.
- ▶ Verification of railway layouts (James, Roggenbach, 2014). Focus on integrating domain modeling (**UML**) with verification, focus on **control systems** and fixed designs.

Controlled natural languages – formally defined restricted subsets of natural language – used for:

- ▶ Object Constraint Language, KeY reasoning about **Java** programs (Johannisson, 2007).
- ▶ **Contract language** \mathcal{CL} (Prisacariu, Schneider, 2012) mapped into natural language and also diagrams (Camilleri, Paganelli, Schneider, 2014).
- ▶ Database queries for **tax fraud detection** (Calafato, Colombo, Pace, 2016).

Overview of approach

- ▶ Define a **Controlled Natural Language** as a high-level **domain-specific** language to write properties.
- ▶ Represent properties as rephrasing of natural language specifications (adds tracability of requirements)

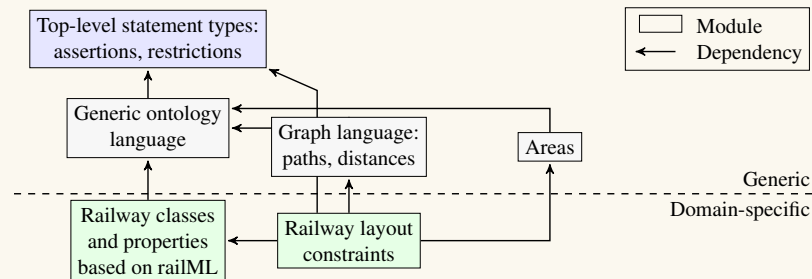


RailCNL: Language design

Top-level statements:

- ▶ **Constraint**: logical constraints, typically used by a Datalog reasoner to infer new facts.
- ▶ **Obligation**: design requirement, CAD model is checked for compliance.
- ▶ **Recommendation**: design heuristics, CAD model checked, but violations are shown as warnings, can be dismissed.

Modules:



RailCNL language design: ontology module

Statements about classes of objects and their properties and relations form a basis for knowledge representation.

- ▶ **Class** names: “*signal*”, “*switch*”, ...
- ▶ **Properties** and **values**: “*color*”, “*red*”, “*200.0m*”, ...
- ▶ **Restrictions**: Equality: “*A signal must have height 4.5m*”.
- ▶ **Relations** name and multiplicity. “*A distant signal should have one or more associated signals.*”

Example 1 (Parse tree for an obligation statement.)

CNL: *A vertical segment must have length greater than 20.0m.*

AST: `OntologyRestriction Obligation
(SubjectClass (StringClassAdjective "vertical"
 (StringClass "segment")))
(ConditionPropertyRestriction (MkPropertyRestriction
 (StringProperty "length")
 (Gt (MkValue (StringTerm "20.0m")))))`

RailCNL language design: graph module

For writing statements about the topology and geometry of objects' placement wrt. to railway tracks.

- ▶ **Goal object:** modifies a subject to optionally add **orientation, direction**, etc.
- ▶ **Path restriction:** combine subject, goal, and path condition. *"All paths from a station border to the first facing switch must pass an entry signal"*.
- ▶ **Distance restriction**, see example:

Example 2 (Parse tree for a railway layout statement.)

CNL: *Distance from an entry signal to first facing switch must be greater than 200.0 m.*

AST: `DistanceRestriction Obligation
 (SubjectClass (StringClassAdjective "entry"
 (StringClass "signal")))
 (FirstFound FacingSwitch)
 (Gt (MkValue (StringTerm "200.0m")))`

Tooling

- ▶ The **quality** of the tool support influences the success of a domain-specific language for non-IT-experts. Textual input is a part of the overall **user interface design**.

Tool support for **RailCNL**:

- ▶ **Paraphrasing** view – present originals and CNL paraphrases side-by-side.
- ▶ **Issues** view – present verification errors in the CAD tool with links to the paraphrasing view.
- ▶ **Editor** – Text editor with support for writing (correct) CNL phrases.

Side-by-side CNL/original (paraphrasing view)

► Requirements tracing

FILE:///home/bj/lut/RailCons/RailCNL/Coverage/Example/skilt-515-plassing.html

TABLE OF CONTENTS

- Hensikt og omfang
- Generelle krav
- Utførelse
- Behandling
- Plassering
- Prosjektering
- Godkjenning/ansvar
- Vedlegg

Hensikt og omfang

De generelle tekniske krav i dette regelverket er et minimum sett av krav til skilt, merker og stolper som skal oppfylles for å ivareta drifts- og personsikkerhet ved alle jernbaneanlegg.

Generelle krav

Utførelse

På jernbaneskilt er det naturlig å skille mellom høyest mulig og en lavere refleksevne. Dette fremgår av tegningen for det enkelte skilt.

For å unngå speilrefleks, bør skilt og merker ikke settes opp vinkelrett (90°) på sporet, men dreies 4° ut.

Behandling

Urtiktig behandling, både under transport og oppsetting, kan føre til skader som nedsetter

ID: skilt1 — Definisjon.

RailCNL: Et skilt har refleksevne høy eller lav.

AST: Constraint (SubjectClass (StringClass "skilt")) (ConditionPropertyRestriction (MkPropertyRestriction (StringProperty "refleksevne") (OrRestr (Eq (MkValue (StringTerm "høy")) (MkValue (StringTerm "lav"))))))

ID: skilt2 — Automatisk verifisering.

RailCNL: Et skilt bør ha sporvinkel som er større enn 94.

AST: Recommendation (SubjectClass (StringClass "skilt")) (ConditionPropertyRestriction (MkPropertyRestriction (StringProperty "sporvinkel") (Gt (MkValue (StringTerm "94"))))))

Datalog:

- skilt2_found(Obj0) :- skilt(Obj0), sporvinkel(Obj0, Val2), Val2 > 94.
- skilt2_recommendation(Obj0) :- skilt(Obj0), !skilt2_found(Obj0).

Issues view

► Backwards tracing – explanation of non-compliance

CAD program showing issues in layout plan



CNL debug view paraphrased text and translations



Original text highlighting source of paraphrased text

Update		
	Category	Description
!	Signal	No interlocking defined.
✘	Signal	The distance from a train detector to another must be greater

ID: detector_1

RailCNL: The distance from an axle counter to another must be larger than 21.0m.

AST: DistanceRestriction Obligation (SubjectClass (StringClassNoAdjective (String "axle_counter"))) (AnyFound (AnyDirectionObject SubjectOtherImplied)) (Gt (MkVal

Datalog: detector_1_start(Subj0, End, Dist) :- trainDetector(Subj0), next(Subj0, End

Placement and length

This section gives generalized rules for placement and length for train detection systems and its relationship to other infrastructure components. Detailed requirements are given in appendices.

General

- No detection sections shall be shorter than 21 meters.**
- No dead zone shall be longer than 3 meters.

Text editor CNL support

- ▶ Rule authoring tool – syntax checks, predictive parsing, chunked parsing, **language exploration**

CloseSubject
Obligation
MkArea

et signal må være plassert i tunnel som har lengde som er større enn 55.0m

StringClassNoAdjective
PlacementRestriction
StringProperty

⇔ Restriction

- AndRestr (-4)
- Eq (0)
- Gte (-1)
- Lt (-1)
- Lte (-1)
- Neq (-1)
- OrRestr (-2)

Advantages

RailCNL as a front-end for property input for verification:

- ▶ RailCNL is **domain-specific**: tailored to Datalog logic and regulations terminology. Gives **readability** and **maintainability**.
- ▶ Resembles **natural language** – improves **readability** and engineer **participation**.
- ▶ Separate textual explanation (such as comments used in programming) are typically not needed.
- ▶ RailCNL statements are **linked** the original text. so that reading them side by side reveals to domain experts whether the CNL paraphrasing of the natural text is valid. If not, they can edit the CNL text.

Further challenges and future work

Participatory verification:

- ▶ RailCNL is a common language **shared** between programmers and railway engineers for verification work.
- ▶ CNLs are not a magical solution to end-user programming.
- ▶ DSLs **evolve** along-side the application.

Language:

- ▶ Structures in regulations that span several phrases/rules (**scopes, exceptions**) – represent on textual or GUI level?
- ▶ Macros – can users extend the language within the scope of their texts?

Tool support:

- ▶ Can railway engineers from other disciplines create their properties themselves, from scratch, with editor support?
- ▶ Is example-based and editor-supported language learning good enough?

Coverage

Classification for coverage analysis:

- ▶ **Not relevant for verification**, examples:

Non-normative: *the technical qualities of the track construction ensure safe and efficient traffic, with the least possible environmental impact.*

Non-checkable: *the tracks' construction must take into account the topography, soil, hydrology, climate, etc. of the location.*

- ▶ **Out of scope** for **static** analysis, examples:

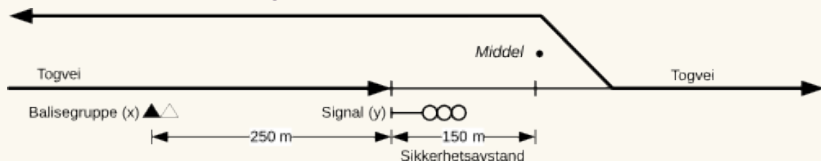
Construction: *Signs must have their original wrapping during transportation.*

Operation: *A signal which cannot signal "stop" because of fault must be unlit.*

Coverage

► Not covered:

- exceptions (awkward to write out all premises)
- linguistically complex: *The safety zone (overlap) can be reduced to 200 m if the speed control system is designed such that the velocity at balise group (x) is not higher than 40 km/h when the signal (y) shows a "stop" aspect, and rolling stock will stop before the fouling point even when speed control communication has failed in both the balise group and in the main signal.*



► Covered:

- ontology, graph, areas, interlocking (targets), ...

Coverage statistics

Eng. discipline	Chapter title	Phrases	Normative	Relevant	Covered	Coverage
Track	Planning: general technical	140	74	74	70	95%
Track	Planning: geometry	278	157	152	119	78%
Signalling	Planning: detectors	144	106	35	21	60%
Signalling	Planning: interlocking	376	265	130	81	62%
Total		938	602	391	291	74%

Table 1: Coverage evaluation for a subset of Norwegian regulations. *Phrases* of the original text which could be classified as *normative* (i.e. applying some restriction on design) were evaluated for *relevance* to static infrastructure verification. The *coverage* is the percentage of relevant phrases expressible in RailCNL.

Participatory verification: experience from meetings between programmers and railway engineers

Positive:

- ▶ invites engineers to splitting hairs
 - discuss semantics of natural language
 - leads to discussion of interpretation of regulations
- ▶ example-based learning
 - explain and explore language with the editor
 - change names and values / copy-paste coding

Negative:

- ▶ total understanding of language is infeasible
 - extend language: ask for examples, not grammar

Datalog verification

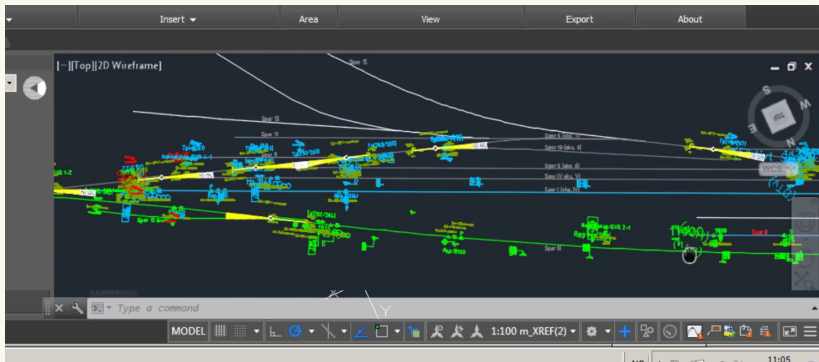
- ▶ Datalog with negation (n.-as-failure) and arithmetic, implemented in e.g. XSB Prolog, RDFox, Soufflé.
- ▶ Prefer very fast (< 100 msec) re-evaluation integrated into CAD tool.
- ▶ Incremental Datalog approaches can exploit locality.

Railway construction process

1. **Politicians** allocate funds for **new** railways, **upgrades** or **maintenance**.
2. National **railway administration** define high level **requirements**, such as passenger/freight **capacities**, **travel times**, maintainability, etc.
3. **Engineering companies** work out the detailed **plans** and **specifications** of the upcoming construction project.
4. **Construction/implementation** companies **build** the railway and **implement** control systems.
5. Finally, **train companies** can transport **passengers** and goods.

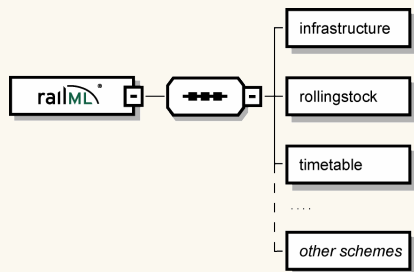
CAD programs in railway signalling

- Overview of a station, typically showing **tracks** and **signalling** system components (signals, signs, balises)



The railML XML standard data exchange format

- ▶ Thoroughly modelled infrastructure schema
- ▶ XML schema development by international standard committee



```
<tracks>
  <track id="tr0" name="01">
    <trackTopology>
      <trackBegin id="x399" pos="0.000000" absPos="346" />
      <connection id="co399" ref="co397" />
    </trackBegin>
    <trackEnd id="y151" pos="80.000000" absPos="346" />
    <connection id="co151_2" ref="co151_1" />
    </trackEnd>
  </trackTopology>
  <trackElements>
    <speedChanges>
      <speedChange id="spu399" pos="0.000000" absPos="346" />
      <speedChange id="spd403" pos="30.000000" absPos="346" />
      <speedChange id="spu405" pos="30.000000" absPos="346" />
      <speedChange id="spd151" pos="80.000000" absPos="346" />
    </speedChanges>
    <gradientChanges>
      <gradientChange id="gr399" pos="0.000000" absPos="346" />
    </gradientChanges>
    <radiusChanges>
      <radiusChange id="ra399" pos="0.000000" absPos="346" />
    </radiusChanges>
    <platformEdges>
      <platformEdge id="pe399" pos="0.000000" absPos="346" />
    </platformEdges>
  </trackElements>
  <ocsElements>
    <signals>
      <signal id="si399" pos="0.000000" absPos="346" />
    </signals>
  </ocsElements>
  </track>
</tracks>
```

Datalog

- ▶ Basic Datalog: conjunctive queries with fixed-point operators (“SQL with recursion”)
 - Guaranteed **termination**
 - **Polynomial** running time (in the number of facts)
- ▶ Expressed as logic programs in a Prolog-like syntax:

$$a(X, Y) :- b(X, Z), c(Z, Y)$$

↕

$$\forall x, y : ((\exists z : (b(x, z) \wedge c(z, y))) \rightarrow a(x, y))$$

- ▶ We also use:
 - Stratified **negation** (negation-as-failure semantics)
 - Arithmetic (which is “unsafe”)