

Tutorial on User-friendly KR

Simon Vandeveld (https://simonvandeveld.be), 11 Nov 2025

Introduction

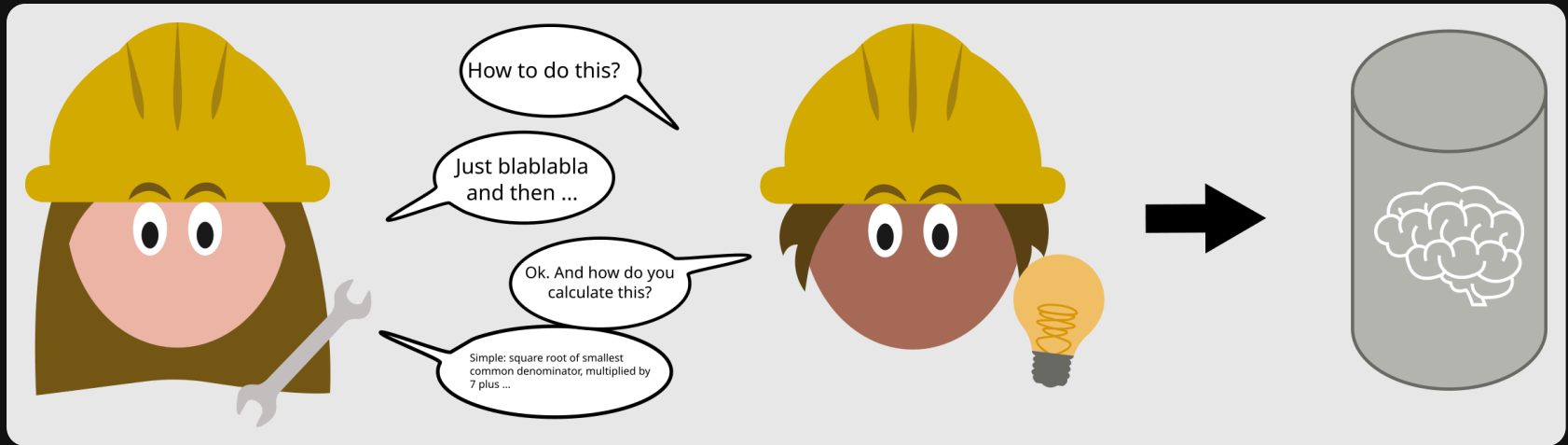
- Post-doc at KU Leuven, Belgium
- *Applied* KR Research:
 - How do we cross the gap between KR research and industry?
 - What struggles do companies experience in adopting KR techniques?
 - How to best support companies?
 - ...

→ worked on many use cases

⚠ Main challenge

Knowledge Acquisition

Classic Knowledge Extraction



Classic Knowledge Extraction

KA is:

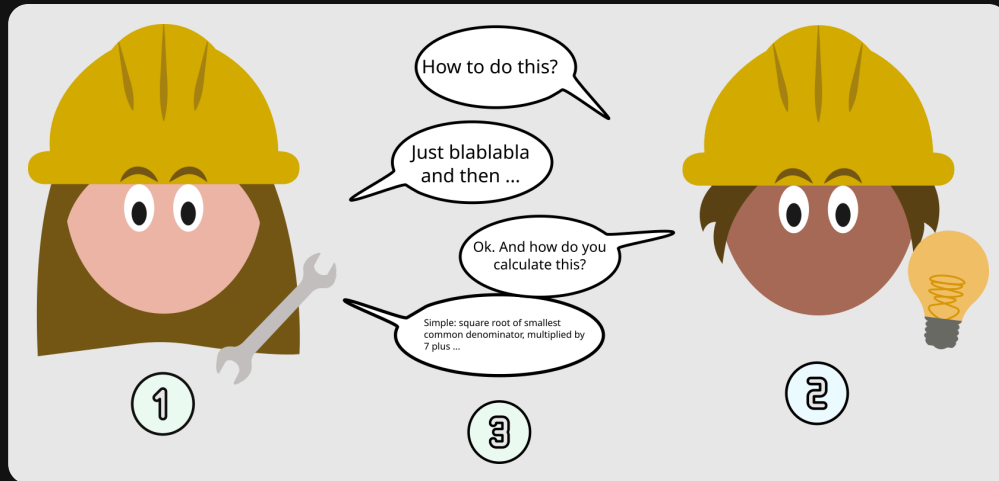
- Time-consuming
- Costly
- Error-prone

→ Main cause: **Both parties speak a different "language"**

① Knowledge Acquisition Bottleneck

The KA Bottleneck has been described since the inception of KR systems.

Knowledge Acquisition Bottleneck



1. Domain expert has knowledge but cannot formalize it
2. Knowledge engineer knows how to formalize but does not understand domain
3. Domain expert and knowledge engineer (implicitly) make assumptions



What if domain expert could model knowledge themselves?

Accessible KR

KR techniques that:

- Are easy to pick up
- Applicable to real-life situations
- Feel *intuitive* for domain experts

This tutorial

Three accessible, "end-user" KR aspects/techniques



Formalisms



Validation



Role of LLMs

Schedule

- 9:00-9:15: Introduction
- 9:15-10:30: 🖋️ Formalisms
- 10:30-11:00: Break
- 11:00-11:30: ✅ Validation
- 11:30-12:00: 🤖 Role of LLMs
- 12:00-12:30: Discussion
- 🍔 Lunch! :-)

Tutorial

This is a *tutorial*: hands-on exercises will be included

- Fully browser-based
- No need to install anything
- Ask questions if you get stuck

Optional: small audience introduction

- What interests you about the topic?
- What would you like to learn?
- Anyone here working on accessible KR?
- ...



Formalisms

User-oriented formalisms

Outline

 **Formalisms**

 Validation

 Role of LLMs

(warning: slight LP bias)

Traditional KR formalisms

- First Order Logic
- Prolog
- ASP
- ...

→ Are these user-friendly?



What makes a notation user-friendly?

User-friendliness is relative

« Beauty is in the eye of the beholder »

« User-friendliness is in the eye of the beholder »

⚠ Beware

We as computer scientists/logicians/... have difficulties estimating this

So what is user-friendly?

Difficult to answer, many facets

Depends heavily on two main factors:



People

- Background
- Type of thinking
- Language
- ...



Problem Domain

- Precise? (e.g., engineering)
- Room for interpretation? (e.g., law)
- Knowledge written down?
- ...

First Order Logic

$$\forall x, y \in Node : edge(x, y) \Rightarrow color(x) \neq color(y)$$

- ● expressive (?)
- ● has a natural reading
- ● complex for non-experts
- ● steep learning curve

Easier for people with some type of maths background

Decision Model and Notation

U	BMI	HealthLevel
1	< 18.5	Underweight
2	[18.5, 25]	Normal
3	> 25	Overweight

- Formalism for decision modeling
- Simple: only 0-ary constants and booleans
- Straightforward reading as if-then
- Graphical, tabular format
- Wide industry adoption

Decision Model and Notation

U	BMI	HealthLevel
1	< 18.5	Underweight
2	[18.5..25]	Normal
3	> 25	Overweight

→ very straightforward FOL semantics!

$$\begin{aligned} & (BMI() < 18.5 \Rightarrow HealthLevel() = Underweight) \\ & \wedge (18.5 \leq BMI() \leq 25 \Rightarrow HealthLevel() = Normal) \\ & \wedge BMI() > 25 \Rightarrow HealthLevel() = Overweight) \end{aligned}$$

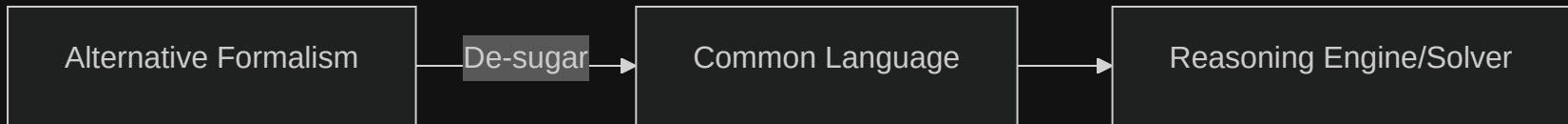


Let's use this as a 'user-facing' formalism!

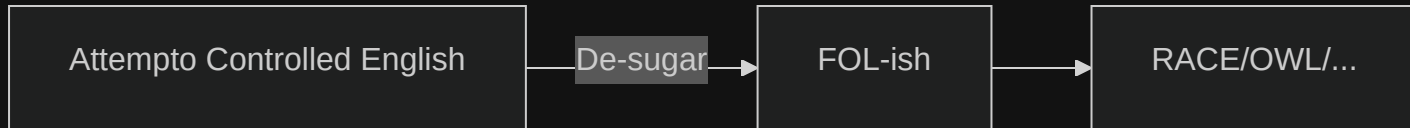
Alternative formalisms

Use formalisms that:

1. Are more accessible
2. Can be translated to common KR language (FOL, ASP, ...)
3. (Ideally) are already in use, have been proven effective



This is not a new idea: e.g., Controlled Natural Languages (CNL)



Exercise: Decision Model and Notation

Let's learn some DMN!

Practically:



IDP-Z3 is a reasoning engine for FO(\cdot), which extends classical FOL

- Types, aggregates, ...
- We'll be working in the web environment
- No need to learn FO(\cdot) which is the entire point :-)

<https://www.omg.org/dmn/>

Calvanese et al. (2019). Semantic DMN: Formalizing and reasoning about decisions in the presence of background knowledge. TPLP

Vandeveldt et al. (2021). Leveraging the power of IDP with the flexibility of DMN: a multifunctional API. RuleML+RR 2021

DMN: quick overview

- Each table *defines* a symbol
- Inputs: green
- Output: blue
- Rows are loosely read as "if-then"

U	BMI	Waist Size	RiskLevel
1	< 18.5	< 90	Increased
2	< 18.5	>= 90	Low
3	[18.5..25]	-	Low
3	> 25	< 90	Increased
4	> 25	>= 90	High

Handyman Exercise

Exercise

Model following knowledge in DMN

A local handyman is an expert at hanging objects on walls. Sometimes he uses a nail, sometimes he uses glue, sometimes he uses a screw. Over the years, he has gathered the following knowledge:

1. Decide if you can make a hole in the wall. This is possible in Brick or Wood walls, but not in Tile
2. Decide how to hang the object:
 - If you can make a hole and the object weighs $\leq 20\text{kg}$, use nails.
 - If you can make a hole and the object weighs $> 20\text{kg}$ and $\leq 40\text{kg}$, use screws.
 - If you can't make a hole and the object weighs $\leq 15\text{kg}$, use glue.



https://slides.simonvandevelde.be/KR25_UserfriendlyKR_Tutorial/

Let's run the model!




→ Run solution in IDP-Z3

→ Or use Interactive Consultant

Handyman: update

Our handyman now has a new tool in his toolkit:

- using a diamond drillbit, he can now drill through tile.

 Exercise

Update the model to reflect this change

Evaluation of DMN

What do you think about DMN?



People

- Tables are intuitive format
- Can work in Excel
- Clear link to rules



Problem Domain

- Works well in domains consisting of rules
 - Law / regulations
 - Compliancy
 - Business logic

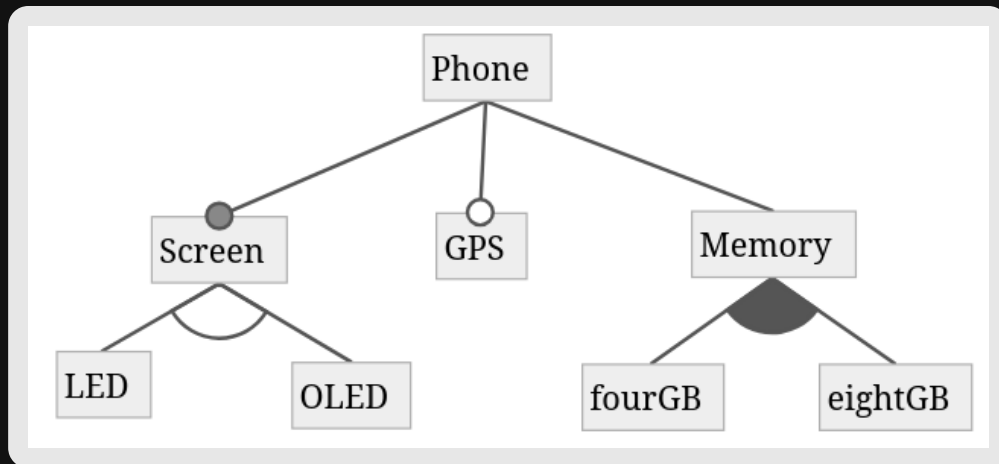
→ However: limited expressiveness

→ Not everything is expressed intuitively in tables

Next Notation: Feature Models

A feature model is a tree representation for configuration problems

- *features* express components
- features have relations over children:
 - Optional, mandatory, OR, alternative
- also cross-tree *include* and *exclude* constraints



Sandwich design

Exercise

Model the following as a feature model

We wish to build a model reflecting the construction of sandwiches. A sandwich consists of three main components:

1. Filling
2. Bread
3. (optional) Mayonaise

For the filling, we can use either ham, cheese, or both. For the bread, we must choose between a baguette or a bagel.

Evaluation of Feature Models

What do you think about feature models?



People

- Tree is intuitive
- Visual format makes it more clear
- Makes sense to, e.g., engineers



Problem Domain

- Works well for configuration problems
- Regardless of domain:
 - Manufacturing
 - Engineering
 - Finance (yes!)
 - ...

→ However: limited expressiveness

→ Not everything is expressed intuitively in tree (e.g., a cost for a feature)

Common flaw: expressiveness

Both notations were quite intuitive, but lacked expressiveness

→ They were not designed with KR in mind!



Can we extend them (while keeping intuitive format!)?

Example: DMN

DMN can only express *definitions*:

- Precisely define a value
- Every input requires a corresponding output
- It's not possible to constrain the output

Exercise

Try to express the following:

You are healthy if you BMI is between [18.5...25]. If you wish to go to the army, you must be healthy

U	BMI	Healthy
1	< 18.5	No
2	> 25	No
3	[18.5..25]	Yes

Option 1:

U	Army	Healthy
1	Yes	Yes
2	No	?

Option 2:

U	Healthy	Army
1	No	No
2	Yes	?

cDMN: constraint DMN

cDMN extends with:

- Constraints
- Quantification
- Types
- Higher-ary symbols
- ...

E*	Army	Healthy
1	Yes	Yes

E*	Healthy	Army
1	No	No

→ No need to specify other cases

Handyman with constraints

Previously we had a concrete decision procedure:

2. Decide how to hang the object:

- If you can make a hole and the object weighs $\leq 20\text{kg}$, use nails.
- If you can make a hole and the object weighs $> 20\text{kg}$ and $\leq 40\text{kg}$, use screws.
- If you can't make a hole and the object weighs $\leq 15\text{kg}$, use glue.

This is derived from the following:

- Nails and screws require a hole
- Nails can hold up to 20kg, screws up to 40kg, and glue up to 15kg



Exercise

Rewrite the Handyman example to be descriptive

Optional: advanced cDMN exercise

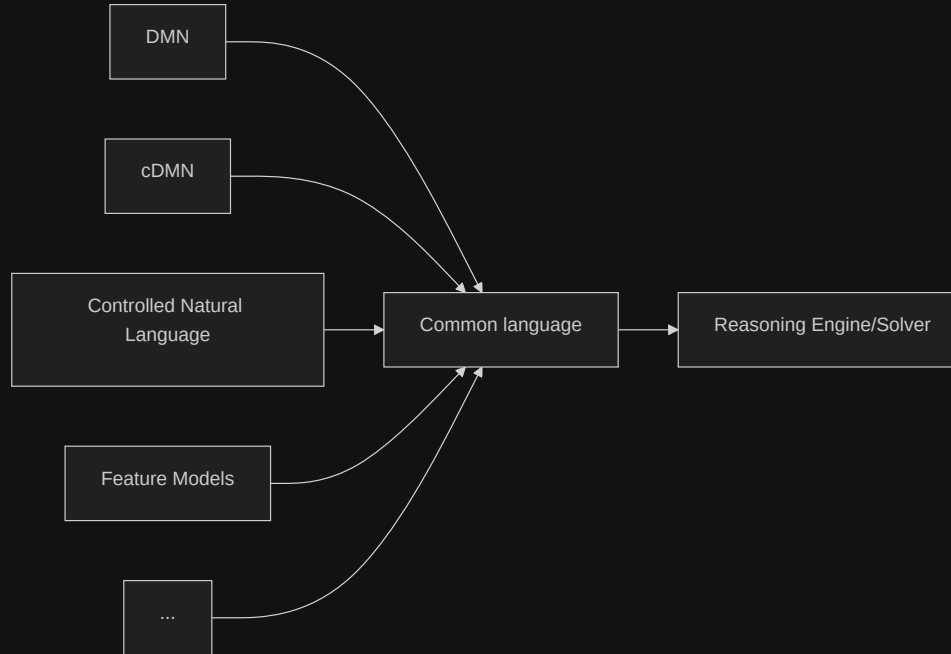
KR-oriented Extensions

In essence:

- Extend existing notations to be better suited for KR
- Maintain intuitive format

→ less steeper learning curve, more accessible

Suite of alternative notations



→ each with different strengths/weaknesses depending on people and domain

Aspects of accessible formalism

- Can be translated to common language
- Aligns no natural intention (e.g., DMN for decision rules)
- Re-uses "commonly known" structures such as trees and tables
- Visual component
- KISS: keep it simple, stupid!

Validation



Outline



Formalisms

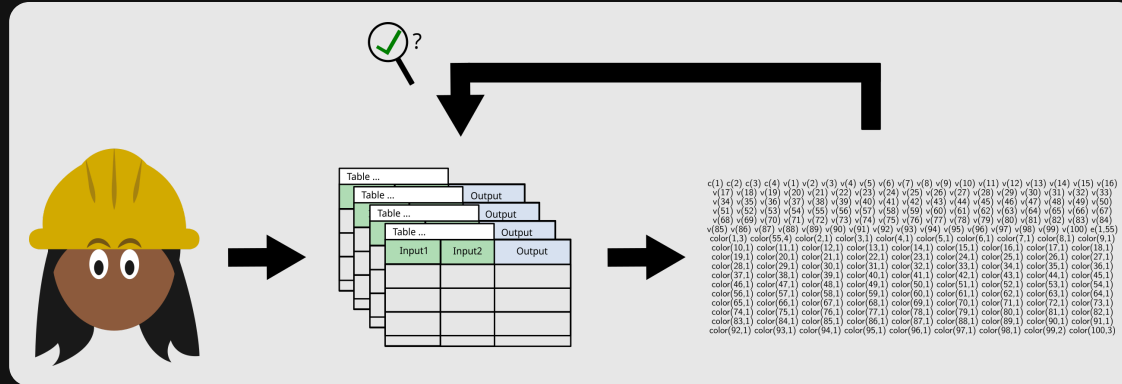


Validation



Role of LLMs

Validation



- Checking correctness
- Modeling correctly is *difficult*!
- I.e., "Does my model correspond to what I intend to express?"

→ output of reasoning engine is not sufficient!

Validation

```
c(1) c(2) c(3) c(4) v(1) v(2) v(3) v(4) v(5) v(6) v(7) v(8) v(9) v(10) v(11) v(12) v(13) v(14) v(15) v(16) v(17) v(18)
v(19) v(20) v(21) v(22) v(23) v(24) v(25) v(26) v(27) v(28) v(29) v(30) v(31) v(32) v(33) v(34) v(35) v(36) v(37) v(38)
v(39) v(40) v(41) v(42) v(43) v(44) v(45) v(46) v(47) v(48) v(49) v(50) v(51) v(52) v(53) v(54) v(55) v(56) v(57) v(58)
v(59) v(60) v(61) v(62) v(63) v(64) v(65) v(66) v(67) v(68) v(69) v(70) v(71) v(72) v(73) v(74) v(75) v(76) v(77) v(78)
v(79) v(80) v(81) v(82) v(83) v(84) v(85) v(86) v(87) v(88) v(89) v(90) v(91) v(92) v(93) v(94) v(95) v(96) v(97) v(98)
v(99) v(100) e(1,55) color(1,3) color(55,4) color(2,1) color(3,1) color(4,1) color(5,1) color(6,1) color(7,1) color(8,1)
color(9,1) color(10,1) color(11,1) color(12,1) color(13,1) color(14,1) color(15,1) color(16,1) color(17,1) color(18,1)
color(19,1) color(20,1) color(21,1) color(22,1) color(23,1) color(24,1) color(25,1) color(26,1) color(27,1) color(28,1)
color(29,1) color(30,1) color(31,1) color(32,1) color(33,1) color(34,1) color(35,1) color(36,1) color(37,1) color(38,1)
.
.
.
```

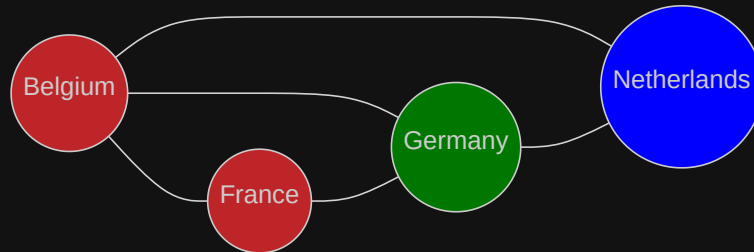
Non-experts should be able to validate:

- quickly
- straightforwardly
- in a manner familiar to them

Visualisations

Visualising solutions can greatly help!

E.g., for a map coloring:



→ Error in the color of France

Caveats:

- Typically requires configuration
- Not all problems are easily visualised

Cliffe, O. et al (2008). ASPVIZ: Declarative Visualisation and Animation Using Answer Set Programming. Logic Programming

Hahn, S., et al. (2024). Clingraph: A System for ASP-based Visualization. TPLP

Kloimüllner, C., et al. (2013). Kara: A System for Visualising and Visual Editing of Interpretations for ASP.

Alviano, M. and Rodriguez Reiners, L. A. (2024). ASP Chef: Draw and Expand. 720–730. <https://doi.org/10.24963/kr.2024/68>

"Explore" problem domain

1. Try out stuff
2. Observe result
3. Check if it's correct
4. Repeat

→ requires interactivity to calculate consequences

Interactivity

We already know such a tool: Interactive Consultant!

Example:

A person may drive if they have either a standard permit or a learner's permit. A standard permit is only possible for 18+, but a learner's permit can already be gotten at 16+.

<https://interactive-consultant.idp-z3.be/?file=permit.idp>

Exercise: validate the model using the IC

Given a model of the following knowledge, check if the knowledge is modeled correctly through the IC:

- If you have more than two symptoms after a risky contact, you must test yourself and quarantine. If you have less, you should test, but not immediately quarantine. If you had no risky contact, you don't need to do anything.
- There are three symptoms to look out for: coughing, sneezing, and fever
- Fever depends on your age: if you are over 10, a body temperature over 38 is considered fever. Otherwise, a body temperature of 37.2 is considered fever.

→ some suggested scenarios in the docs



https://slides.simonvandevelde.be/KR25_UserfriendlyKR_Tutorial/

Large Language Models

Outline



Formalisms



Validation



Role of LLMs

LLMs and reasoning

🧠 Can LLMs reason?

- Not "faithful": hallucinations/confabulations slip in
- Not transparent
- Fundamentally: black-box probabilistic model

Still much debate!^{1,2,3}

🧠 Can reasoning engines reason?



Can we offload reasoning to dedicated reasoning engine?

¹Bender et al. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? FAccT '21.

²Wu et al. 2024. Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks. ACL 2024

³Shojaee et al. 2025. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity,

LLM-based formalization: SotA

Two main groups:

Formalize domain knowledge



Generate model based on NL

- General; no intended reasoning task
- Show potential, not perfect
- Goossens 2023, Ishay 2023, Mensfelt 2024, Coppolillo 2024

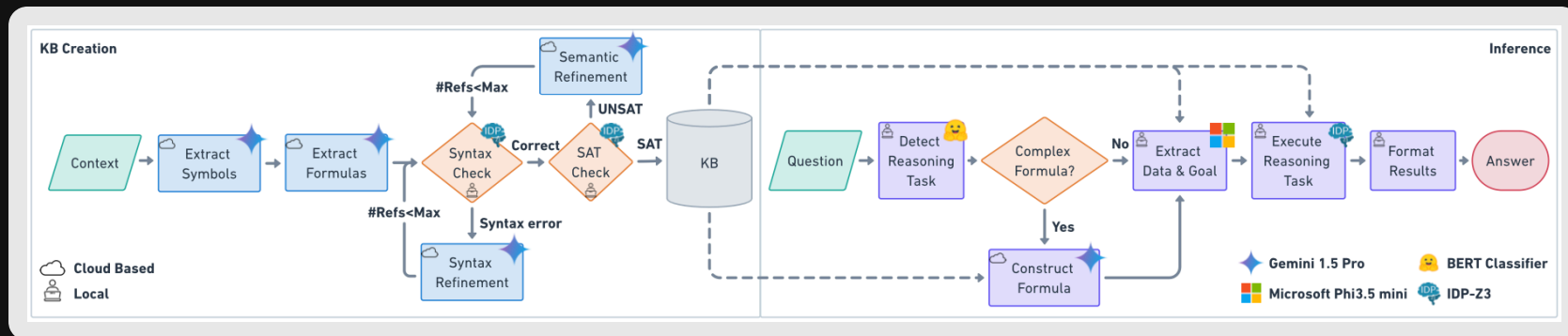
Improve LLM reasoning



'outsource' reasoning task to engine

- Based on description and reasoning task
- Outperform baseline LLMs
- Olausson 2023, Pan 2023, Yang 2023, Callewaert 2025

Example: VERUS-LM



Example: VERUS-LM

Knowledge:

To calculate a patient's BMI, divide their weight by their height squared.

Query:

What is the BMI of a person of 1.79m weighing 80kg?

1. Symbol Extraction:

```
height: -> Int  
weight: -> Int  
BMI: -> Int
```

2. Formula Extraction:

```
BMI() = weight()/(height() * height())
```

Example: VERUS-LM

3. Refinement steps

4. Inference detection:

```
model generation
```

5. Information extraction:

```
structure S {  
  height := 1.79.  
  weight := 80.  
}
```

6. Execute and format output

```
A patient with a height of 1.79 weighing 80kg would have a BMI of 24.96.
```

Holy grail

Automatically build tools by:

1. Handing internal docs to an LLM
2. Letting the LLM formalize a KB
3. Plug the KB into an interface

Holy fail:

- LLMs make formalization errors
- LLMs can still hallucinate/confabulate
- It is difficult to check if the resulting KB is correct

→ If LLMs only achieve 89% accuracy on small problems, how well will they perform on entire documents?

→ Alternative: domain expert in the loop!

Domain expert in the loop

- With the current LLMs, auto-formalization seems impossible
- In human-human KA, domain experts are crucial for validation!
- Why should this be different when using LLM?
- Still, domain experts cannot interpret formal models directly



How can a domain expert independently validate a KB?

→ Same ideas as previously:

1. Visualisation and interaction tools
2. End-user formalisms
3. Incremental formalization

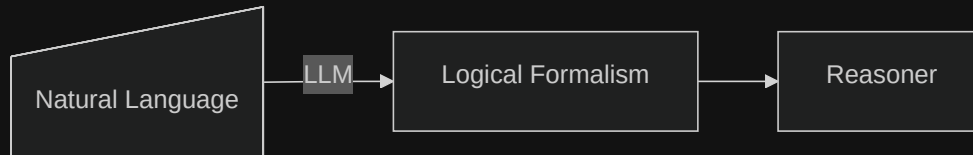
(More are possible)

Visualisation and interaction tools

- Visualise one or more solutions
- Interactively explore problem domain
- Verify that the model's behavior matches the expectations

End-user formalism

Current approach:



User cannot understand produced format.

What if we could translate into an alternative notation instead?



End-user formalism

Idea: intermediary formalism that is

- More intuitive for non-experts
- Directly translatable into "traditional" formal logic

The barrier for validating such statements would be much lower.

Well-known example: Controlled Natural Languages, e.g., ACE:

```
Every country is a territory.  
If X borders Y then Y borders X.  
If X borders something then X is a country.  
Germany borders Switzerland.
```

```
!x: country(x) => territory(x).  
!x, y: borders(x, y) => borders(y, x).  
!x, y: borders(x, y) => borders(y, x).  
!x, y: borders(x, y) => country(x).  
borders(Germany, Switzerland).
```


End-user formalisms

Multiple examples of such CNLs:

- ACE (Fuchs 2008)
- PENG (White 2009)
- CNL2ASP (Caruso 2023)

Other end-user formalisms also exist, but are often more graphical in nature

Extra: Domain-specific formalisms

- Domain-specific notation
- Aligns better to natural intuition of domain expert
- E.g., Logical English for regulatory knowledge

Incremental formalization

- LLM-based formalization focusses on "single-shot" translations
- I.e., a document into KB, instantly
- More difficult to validate
- LLMs are also limited in input tokens!

Instead, we should use *incremental formalization*

- Build model in multiple steps
- Decompose in substeps, iteratively refine model
- Each step allows for "bite-sized" validation

→ Bonus: would allow for "interactive chat sessions" (explain more *tacit knowledge*)

User-centric LLM-based formalization

Three ideas:

1. Visualisation and interaction
2. End-user formalisms
3. Incremental formalization

→ These are not mutually exclusive!

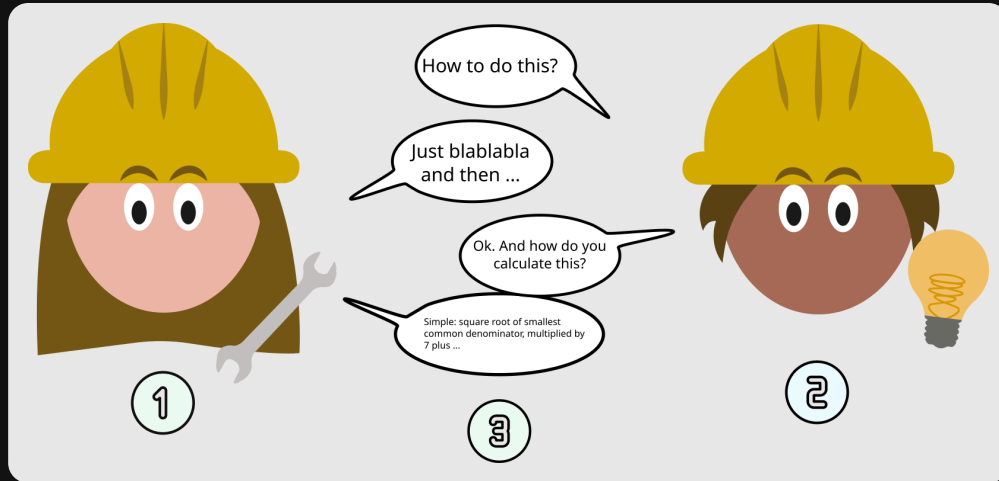
Challenges

- Involving non-experts leads to high variability (e.g., based on formal background)
- Tools *really* need to be user-friendly
- Syntactic and semantic correctness of LLMs
- Incremental formalization is not always straightforward!

Wrap-up



Knowledge Acquisition Bottleneck



1. Domain expert has knowledge but cannot formalize it
2. Knowledge engineer knows how to formalize but does not understand domain
3. Domain expert and knowledge engineer (implicitly) make assumptions



What if domain expert could model knowledge themselves?

Accessible KR

KR techniques that:

- Are easy to pick up
- Applicable to real-life situations
- Feel *intuitive* for domain experts



Formalisms

- Alternative, "end-user" formalism
- User-specific
- Problem-specific



Validation

- Visualisation and interaction
- Explore problem domain
- Abstract



Role of LLMs

- Autoformalization
- Keep domain experts in the loop

Thank you!

① Workshop on End-User Logic Programming (EULP)

FLOC 2026: workshop on all work related to bringing end-users in the loop

Slides: https://slides.simonvandevelde.be/KR25_UserfriendlyKR_Tutorial/



<https://simonvandevelde.be>



s.vandevelde@kuleuven.be



[@saltfactory@mastodon.social](https://mastodon.social/@saltfactory)



[Simon Vandevelde](#)