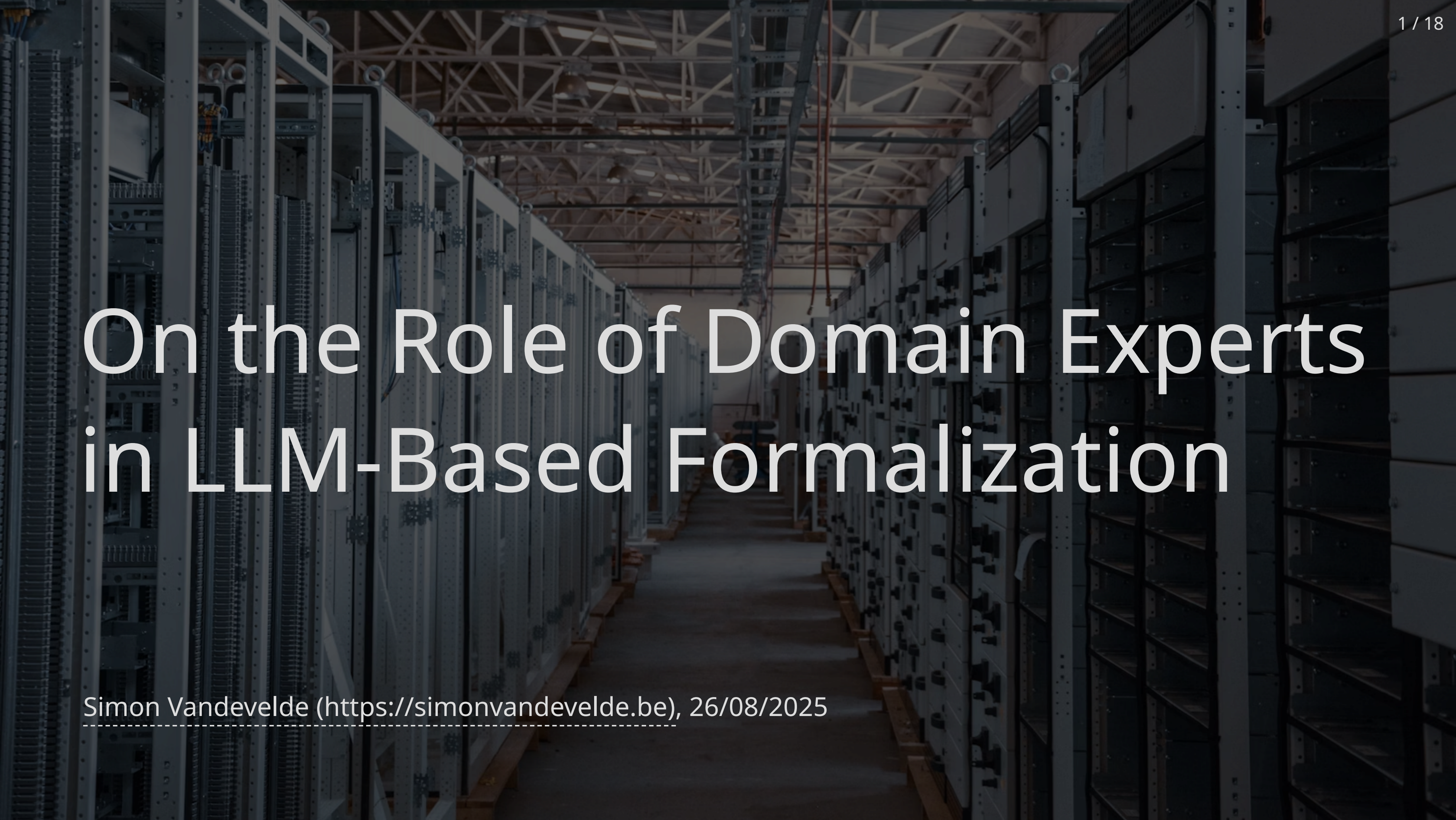


On the Role of Domain Experts in LLM-Based Formalization

Simon Vandevelde (https://simonvandevelde.be), 26/08/2025

Logic-based application

Company has staff specialized in process : tricky, time-consuming, error-prone

they wish to support staff with tool

is very knowledge-intensive

They know that:

✅ Logic-based tools exist

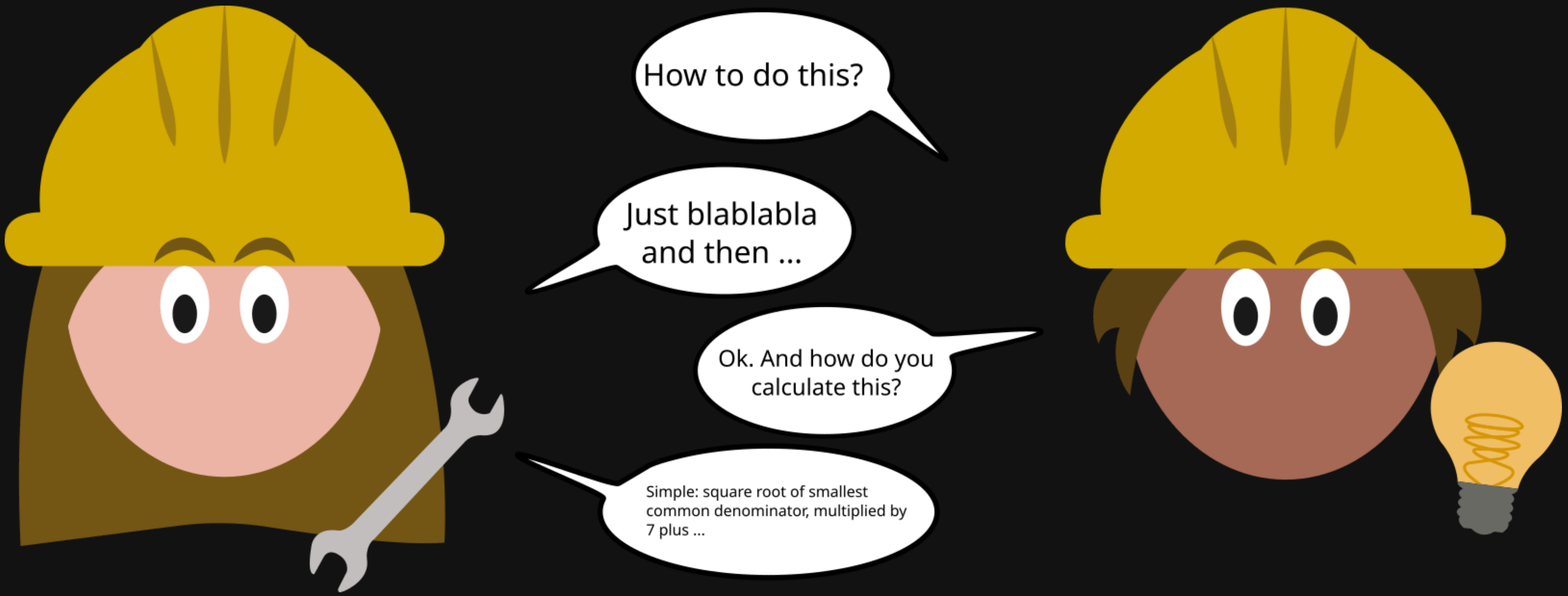
✅ Expressive KR languages

✅ Performant reasoners

❌ How to model knowledge

💭 Let's get experts to help us!

Knowledge Acquisition



Suzy: domain expert
(cannot formalize knowledge)

Ben: knowledge engineer
(does not know the problem domain

Knowledge Acquisition: an issue

Both parties have *very* distinct knowledge, making KA

- Time-consuming
- Error-prone
- Labour-intensive

much time needs to be spent on validation!

KA is notoriously difficult

Knowledge Acquisition Bottleneck

💭 Can we use LLMs to formalize knowledge instead?

LLM-based formalization: SotA

Two main groups:

Formalize domain knowledge

🎯 Generate model based on NL

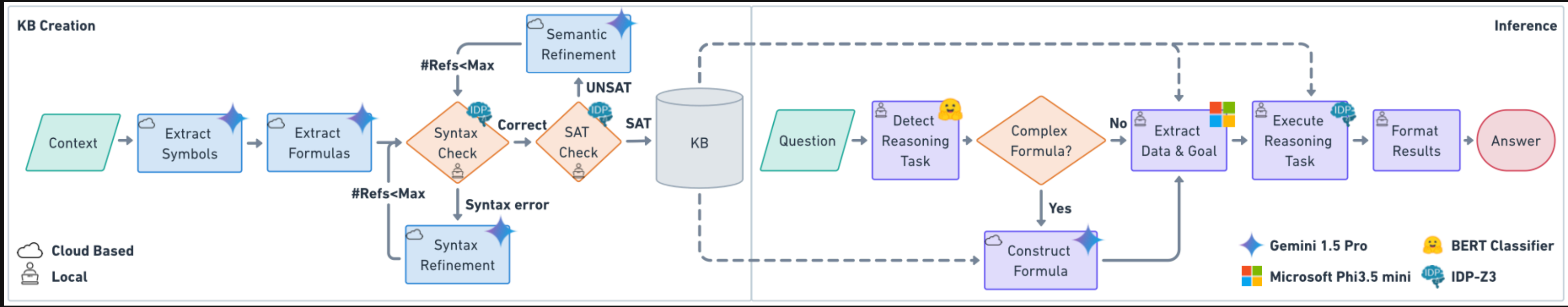
- General; no intended reasoning task
- Show potential, not perfect
- Goossens 2023, Ishay 2023, Mensfelt 2024, Coppolillo 2024

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?

Three ideas:

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

Example: Interactive Consultant

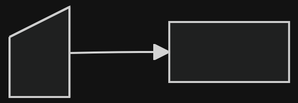
> A person may drive if they have either a standard permit or a learner’s permit. A standard permit is only possible for 18+, but a learner’s permit can already be gotten at 16+.

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

Other tools: clinguin, clingraph, ASP Chef, Clafer configurator

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" where domain expert can explain their 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!

Thank you

Slides: https://slides.simonvandevelde.be/SeminarDjordje/slides.pdf

https://simonvandevelde.be

s.vandevelde@kuleuven.be

@saltfactory@mastodon.social

Simon Vandevelde