May 2025, Les Houches

https://iltis.rub.de

Iltis

Learning Formal Foundations of CS

Thomas Zeume

Ruhr University Bochum

Project coordinators:	Marko Schmellenkamp, Fabian Vehlken, Thomas Zeume
Interface Design:	Christine Orhan
Implementation:	Sven Argo, Jill Berg, Emilio Carrasco Bustamante, Lukas Dienst, Aiyana Erbe, Gaetano Geck, Ari Haji, Jonas Haldimann, Lauin Hamow, Tristan Kneisel, Alexandra Latys, Artur Ljulin, Johannes May, Jan Michalak, Sebastian Peter, Lukas Pradel, Lars Richter, Marius Rößler, Patrick Roy, Florian Schmalstieg, Jonas Schmidt, Cedric Siems, Daniel Sonnabend, Fynn Stebel, Felix Tschirbs, Patrick Wieland, Oskar Wilke, Jan Zumbrink
Contents:	Tom-Felix Berger, Ariane Blank, Jan Eyll, Alicia Gayda, Elias Radtke, Sidra Saied Ali, Florian Schmalstieg, Mira Schwartz, Thomas Schwentick, Cara Volbracht, Marco Wojtek

Part I:

A Short Guided Tour

Learning Logic in the Web

National Research council of the US:

Leverage technologies to make the most effective use of students' time, shifting from information delivery to sense-making and practice in class

Learning Logic in the Web

National Research council of the US:

Leverage technologies to make the most effective use of students' time, shifting from information delivery to sense-making and practice in class

Use technology to

- provide additional learning opportunities for students
- make room for theory and in-depth problem solving in class

Learning Logic in the Web

National Research council of the US:

Leverage technologies to make the most effective use of students' time, shifting from information delivery to sense-making and practice in class

Use technology to

- provide additional learning opportunities for students
- make room for theory and in-depth problem solving in class

Our goals for technological learning support:

- **Coverage** of a wide range of topics in formal foundations of computer science;
- **Advanced feedback and support** provided immediately, extensively, and individually;
- **Flexibility** in how to use and combine educational tasks;
- **Easy integration** into courses; and
- **Extensibility** of topical range and feedback mechanisms.

An Assignment for Propositional Logic (1/2)

An Exercise for Propositional Logic

An analysis has revealed the following dependencies among three components of a software system:

1. If the backend is working correctly, the database is also working correctly.
2. The backend is only working incorrectly if neither the database nor the user interface is working correctly.
3. At least one component works correctly.

Task A: modelling
Provide a propositional formula for each dependency.

Task B: transformation and reasoning
Show by resolution that the dependencies imply that the database and the backend work correctly.

An Assignment for Propositional Logic (1/2)

An Exercise for Propositional Logic

An analysis has revealed the following dependencies among three components of a software system:

1. If the backend is working correctly, the database is also working correctly.
2. The backend is only working incorrectly if neither the database nor the user interface is working correctly.
3. At least one component works correctly.

Task A: modelling
Provide a propositional formula for each dependency.

Task B: transformation and reasoning
Show by resolution that the dependencies imply that the database and the backend work correctly.

(1) Choose a vocabulary

(2) Formalize the scenario

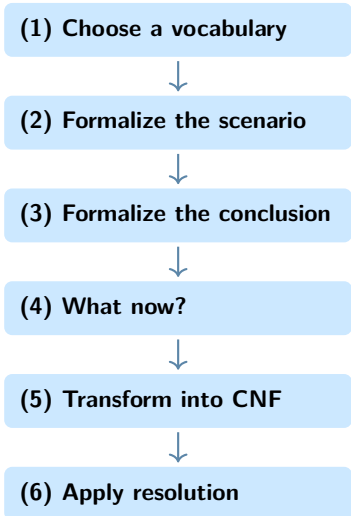
(3) Formalize the conclusion

(4) What now?

(5) Transform into CNF

(6) Apply resolution

An Assignment for Propositional Logic (2/2)



An Assignment for Propositional Logic (2/2)

(1) Choose a vocabulary

↓

(2) Formalize the scenario

↓

(3) Formalize the conclusion

↓

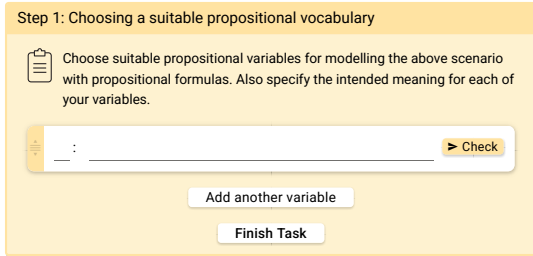
(4) What now?

↓

(5) Transform into CNF

↓

(6) Apply resolution



An Assignment for Propositional Logic (2/2)

(1) Choose a vocabulary

↓

(2) Formalize the scenario

↓

(3) Formalize the conclusion

↓

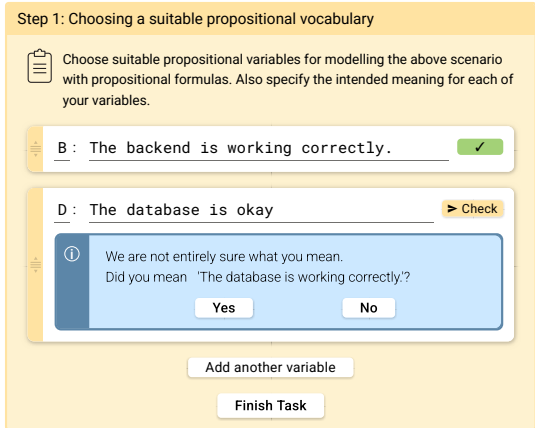
(4) What now?

↓

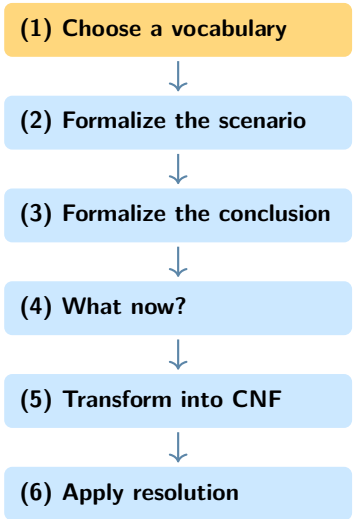
(5) Transform into CNF

↓

(6) Apply resolution



An Assignment for Propositional Logic (2/2)



Step 1: Choosing a suitable propositional vocabulary

Choose suitable propositional variables for modelling the above scenario with propositional formulas. Also specify the intended meaning for each of your variables.

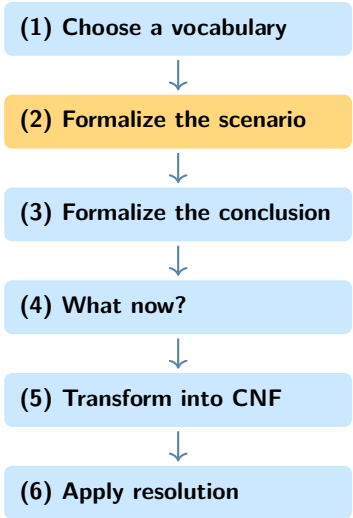
B : The backend is working correctly. ✓

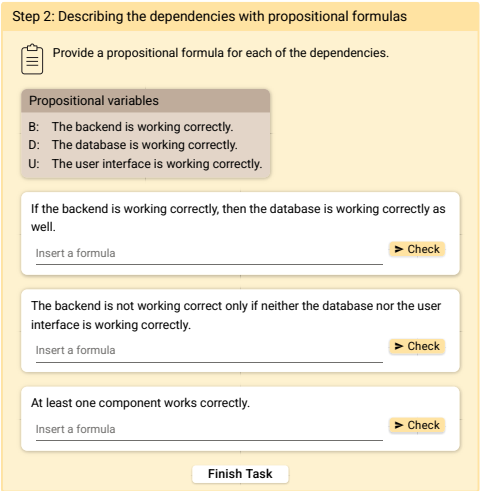
D : The database is working correctly. ✓

U : The user interface is working correctly. ✓

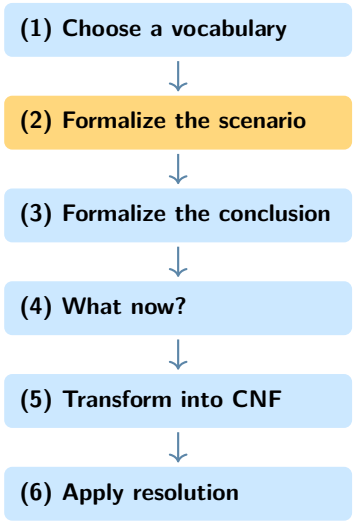
Congratulations! You have successfully completed this task.

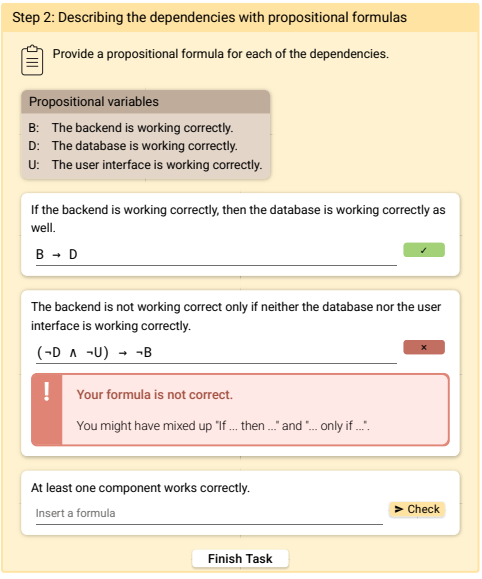
An Assignment for Propositional Logic (2/2)



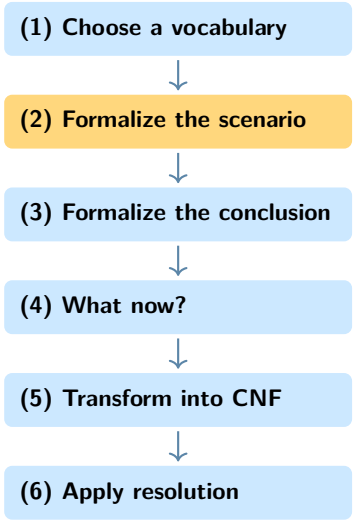


An Assignment for Propositional Logic (2/2)





An Assignment for Propositional Logic (2/2)



Step 2: Describing the dependencies with propositional formulas

Provide a propositional formula for each of the dependencies.

Propositional variables

B: The backend is working correctly.
D: The database is working correctly.
U: The user interface is working correctly.

If the backend is working correctly, then the database is working correctly as well.

`B → D` ✓

The backend is not working correct only if neither the database nor the user interface is working correctly.

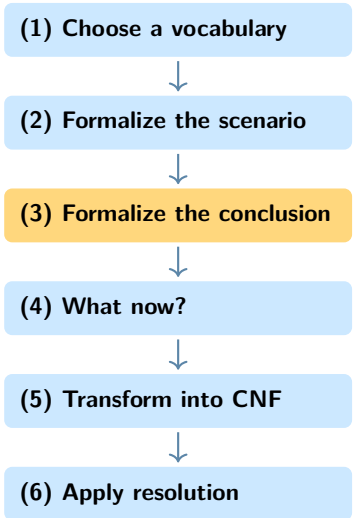
`¬B → (¬D ∧ ¬U)` ✓

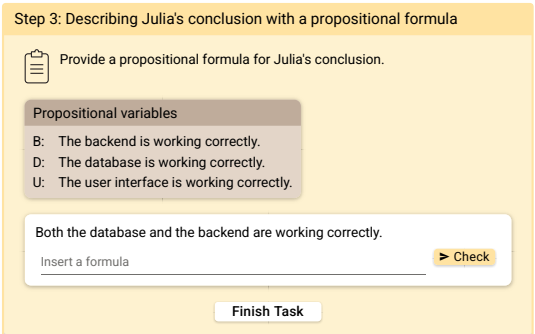
At least one component works correctly.

`B ∨ D ∨ U` ✓

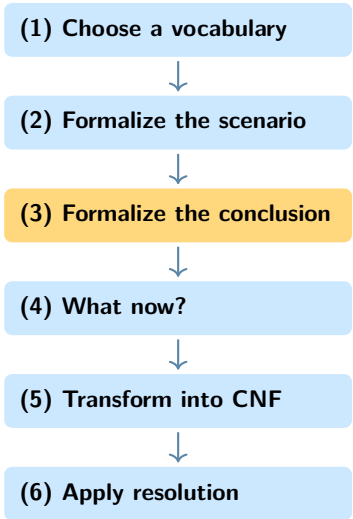
Congratulations! You have successfully completed this task.

An Assignment for Propositional Logic (2/2)





An Assignment for Propositional Logic (2/2)



Step 3: Describing Julia's conclusion with a propositional formula

Provide a propositional formula for Julia's conclusion.

Propositional variables

B: The backend is working correctly.
D: The database is working correctly.
U: The user interface is working correctly.

Both the database and the backend are working correctly.

B ∧ D ✓

Congratulations! You have successfully completed this task.

An Assignment for Propositional Logic (2/2)

(1) Choose a vocabulary

↓

(2) Formalize the scenario

↓

(3) Formalize the conclusion

↓

(4) What now?

↓

(5) Transform into CNF

↓

(6) Apply resolution

Step 4: What to prove?

Our goal is to show that Julia's conclusion can be inferred from the observed dependencies. We have already modelled the dependencies between the individual components by the three propositional formulas

φ_1: If the backend is working correctly, the database is also working correctly.
φ_2: The backend is not working correct only if neither the database nor the user interface is working correctly.
φ_3: At least one component works correctly.

and Julia's conclusion by the formula

ψ: Both the database and the backend are working correctly.

What do we need to show in order to verify that Julia's conclusion follows from the dependencies of the software system?

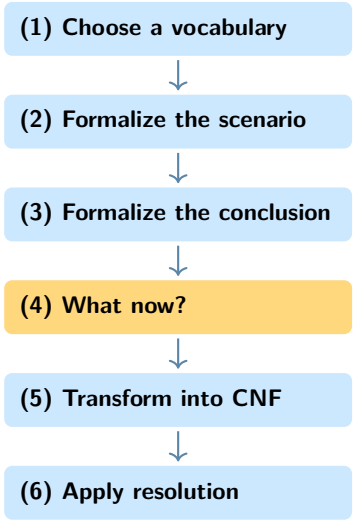
Answer this question by selecting all correct answers.

Julia's conclusion holds if and only if . . .

- ☐ the formula ψ has a model.
- ☐ the inference $\psi \models \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ holds.
- ☐ the inference $\{\varphi_1, \varphi_2, \varphi_3\} \models \psi$ holds.
- ☐ an assignment α with $\alpha \models \psi$ exists.
- ☐ the formula ψ can be inferred from the formula $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$.
- ☐ every assignment compatible with $\varphi_1, \varphi_2, \varphi_3$ and ψ which is a model of $\{\varphi_1, \varphi_2, \varphi_3\}$ is also a model of ψ .

Submit answers

An Assignment for Propositional Logic (2/2)



Step 4: What to prove?

Our goal is to show that Julia's conclusion can be inferred from the observed dependencies. We have already modelled the dependencies between the individual components by the three propositional formulas

- φ_1: If the backend is working correctly, the database is also working correctly.
- φ_2: The backend is not working correct only if neither the database nor the user interface is working correctly.
- φ_3: At least one component works correctly.

and Julia's conclusion by the formula

- ψ: Both the database and the backend are working correctly.

What do we need to show in order to verify that Julia's conclusion follows from the dependencies of the software system?

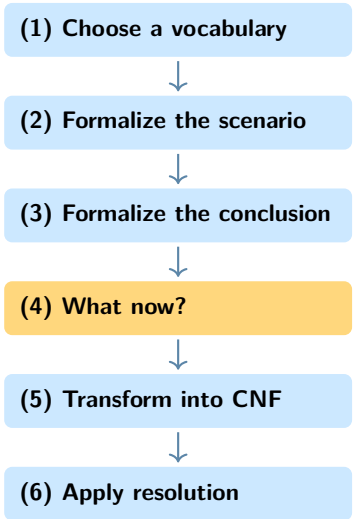
Answer this question by selecting all correct answers.

Julia's conclusion holds if and only if . . .

- ☐ the formula ψ has a model.
- ☐ the inference $\psi \models \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ holds.
- ☑ the inference $\{\varphi_1, \varphi_2, \varphi_3\} \models \psi$ holds.
- ☐ an assignment α with $\alpha \models \psi$ exists.
- ☑ the formula ψ can be inferred from the formula $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$.
- ☑ every assignment compatible with $\varphi_1, \varphi_2, \varphi_3$ and ψ which is a model of $\{\varphi_1, \varphi_2, \varphi_3\}$ is also a model of ψ .

Congratulations! You have successfully completed this task.

An Assignment for Propositional Logic (2/2)



Step 4: How to prove the inference?

In order to verify that Julia's conclusion is correct, we need to prove the inference. How can we prove this?

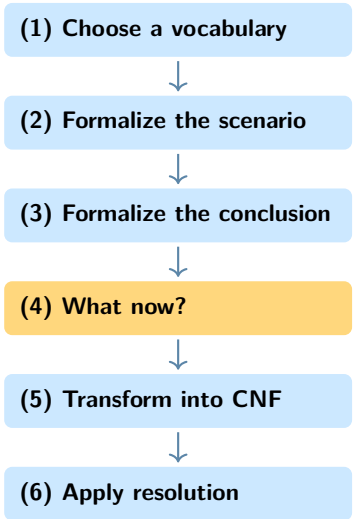
Answer this question by selecting all correct answers.

The inference $\{\varphi_1, \varphi_2, \varphi_3\} \models \psi$ holds if and only if ...

- ☐ $\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \neg\psi$ has a model.
- ☐ $\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \neg\psi$ is unsatisfiable.
- ☐ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi$ has a model.
- ☐ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi$ is unsatisfiable.
- ☐ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi$ is a tautology.
- ☐ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi \equiv \bot$ holds.
- ☐ $(\varphi_1 \wedge \varphi_2 \wedge \varphi_3) \rightarrow \psi \equiv \top$ holds.
- ☐ $(\varphi_1 \wedge \varphi_2 \wedge \varphi_3) \rightarrow \psi \equiv \bot$ holds.

Submit answers

An Assignment for Propositional Logic (2/2)



Step 4: How to prove the inference?

In order to verify that Julia's conclusion is correct, we need to prove the inference. How can we prove this?

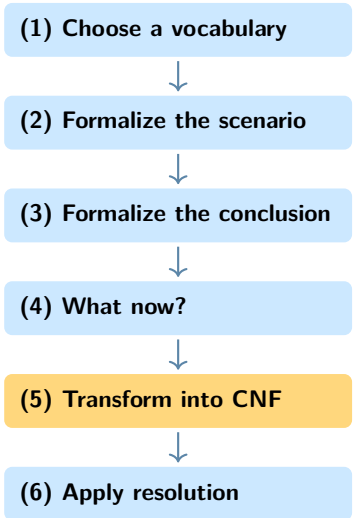
Answer this question by selecting all correct answers.

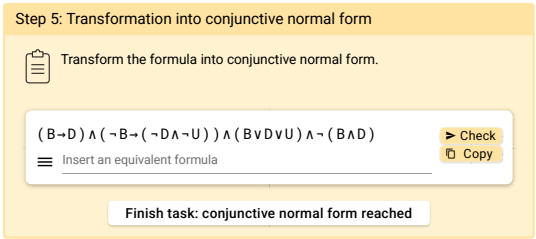
The inference $\{\varphi_1, \varphi_2, \varphi_3\} \models \psi$ holds if and only if ...

- ☐ $\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \neg\psi$ has a model.
- ☐ $\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \neg\psi$ is unsatisfiable.
- ☐ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi$ has a model.
- ☑ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi$ is unsatisfiable.
- ☐ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi$ is a tautology.
- ☑ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\psi \equiv \bot$ holds.
- ☑ $(\varphi_1 \wedge \varphi_2 \wedge \varphi_3) \rightarrow \psi \equiv \top$ holds.
- ☐ $(\varphi_1 \wedge \varphi_2 \wedge \varphi_3) \rightarrow \psi \equiv \bot$ holds.

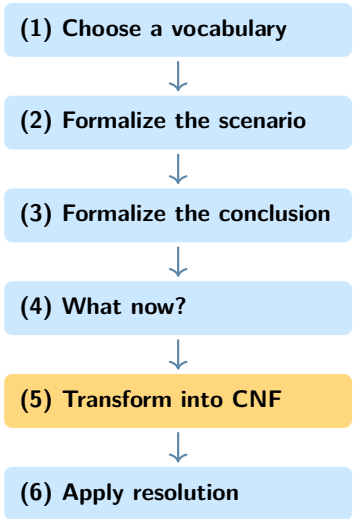
Congratulations! You have successfully completed this task.

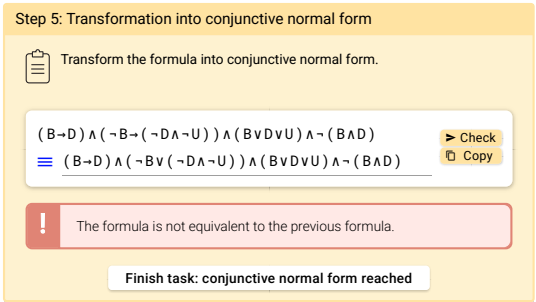
An Assignment for Propositional Logic (2/2)



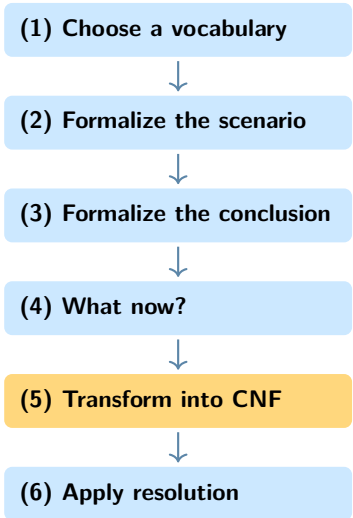


An Assignment for Propositional Logic (2/2)





An Assignment for Propositional Logic (2/2)



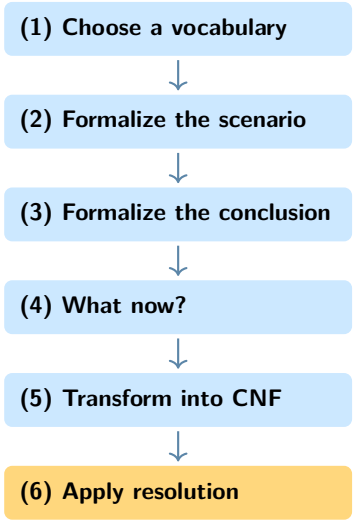
Step 5: Transformation into conjunctive normal form

Transform the formula into conjunctive normal form.

$$
\begin{aligned}
&(B \to D) \wedge (\neg B \to (\neg D \wedge \neg U)) \wedge (B \vee D \vee U) \wedge \neg (B \wedge D) \\
\equiv\ &(\neg B \vee D) \wedge (\neg \neg B \vee (\neg D \wedge \neg U)) \wedge (B \vee D \vee U) \wedge \neg (B \wedge D) \\
\equiv\ &(\neg B \vee D) \wedge (B \vee (\neg D \wedge \neg U)) \wedge (B \vee D \vee U) \wedge (\neg B \vee \neg D) \\
\equiv\ &(\neg B \vee D) \wedge (B \vee \neg D) \wedge (B \vee \neg U) \wedge (B \vee D \vee U) \wedge (\neg B \vee \neg D)
\end{aligned}
$$

Congratulations! You have successfully completed this task.

An Assignment for Propositional Logic (2/2)



Step 6: Propositional resolution

Now use propositional resolution to show that the set of clauses resulting from the formula above is unsatisfiable.

Resolve

$\{\neg B, D\}$ $\{B, \neg D\}$ $\{B, \neg U\}$ $\{B, D, U\}$ $\{\neg B, \neg D\}$

Finish task: empty clause reached

An Assignment for Propositional Logic (2/2)

(1) Choose a vocabulary

↓

(2) Formalize the scenario

↓

(3) Formalize the conclusion

↓

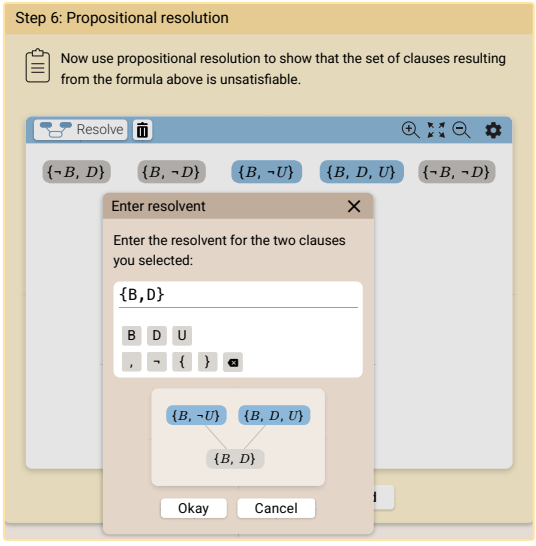
(4) What now?

↓

(5) Transform into CNF

↓

(6) Apply resolution



An Assignment for Propositional Logic (2/2)

(1) Choose a vocabulary

↓

(2) Formalize the scenario

↓

(3) Formalize the conclusion

↓

(4) What now?

↓

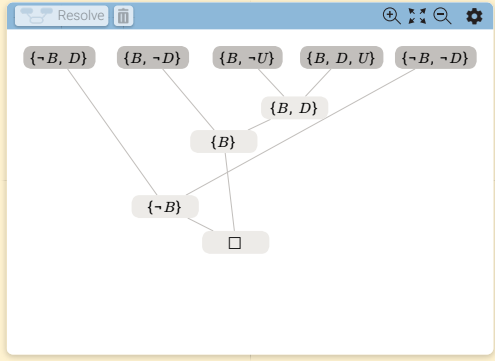
(5) Transform into CNF

↓

(6) Apply resolution

Step 6: Propositional resolution

Now use propositional resolution to show that the set of clauses resulting from the formula above is unsatisfiable.



Congratulations! You have successfully completed this task.

An Assignment for First-order Logic

A First-order Exercise

Julia has identified dependencies between her software packages:

(1) No software package is both a program library and a system library.

. . .

(4) Every software package that must be explicitly installed by the user depends on at least one program library directly.

Task: modelling
Provide a first-order formula for each dependency.

An Assignment for First-order Logic

A First-order Exercise

Julia has identified dependencies between her software packages:

(1) No software package is both a program library and a system library.

. . .

(4) Every software package that must be explicitly installed by the user depends on at least one program library directly.

Task: modelling
Provide a first-order formula for each dependency.

(1) Choose a vocabulary

(2) Formalize the scenario

An Assignment for First-order Logic

A First-order Exercise

Julia has identified dependencies between her software packages:

(1) No software package is both a program library and a system library.

. . .

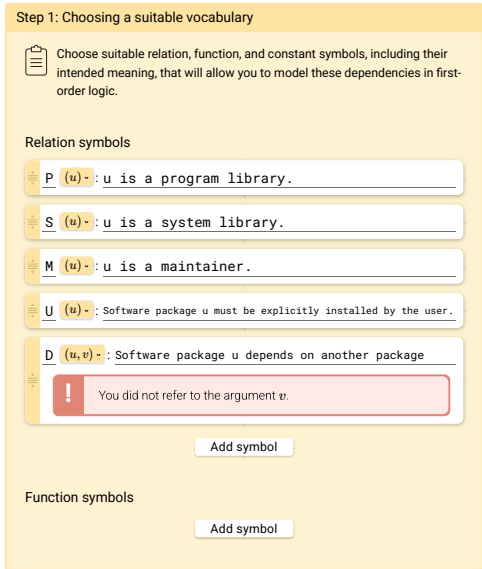
(4) Every software package that must be explicitly installed by the user depends on at least one program library directly.

Task: modelling
Provide a first-order formula for each dependency.

(1) Choose a vocabulary

↓

(2) Formalize the scenario



An Assignment for First-order Logic

A First-order Exercise

Julia has identified dependencies between her software packages:

(1) No software package is both a program library and a system library.

. . .

(4) Every software package that must be explicitly installed by the user depends on at least one program library directly.

Task: modelling
Provide a first-order formula for each dependency.

(1) Choose a vocabulary

↓

(2) Formalize the scenario

Step 1: Choosing a suitable vocabulary

Choose suitable relation, function, and constant symbols, including their intended meaning, that will allow you to model these dependencies in first-order logic.

Relation symbols

- P (u) : `u is a program library.`
- S (u) : `u is a system library.`
- M (u) : `u is a maintainer.`
- U (u) : `Software package u must be explicitly installed by the user.`
- D (u, v) : `Software package u depends on software package v.`

Function symbols

- m (u) : `Maintainer of software package u`

Congratulations! You have successfully completed this task.

An Assignment for First-order Logic

A First-order Exercise

Julia has identified dependencies between her software packages:

(1) No software package is both a program library and a system library.

. . .

(4) Every software package that must be explicitly installed by the user depends on at least one program library directly.

Task: modelling
Provide a first-order formula for each dependency.

(1) Choose a vocabulary

↓

(2) Formalize the scenario

Step 2: Describing the dependencies with first-order formulas

Your vocabulary

$P(u)$:	u is a program library.
$S(u)$:	u is a system library.
$M(u)$:	u is a maintainer.
$U(u)$:	Software package u must be explicitly installed by the user.
$D(u, v)$:	Software package u depends on software package v.
$m(u)$:	Maintainer of software package u

No software package is both a program library and a system library.

`∃u ¬(P(u) ∧ S(u))`

! Your formula is not correct.

Show more...

System libraries depend only on system libraries.

Insert a formula

Program libraries may only depend on system libraries which are not maintained by the same maintainer.

Insert a formula

An Assignment for First-order Logic

A First-order Exercise

Julia has identified dependencies between her software packages:

(1) No software package is both a program library and a system library.

. . .

(4) Every software package that must be explicitly installed by the user depends on at least one program library directly.

Task: modelling
Provide a first-order formula for each dependency.

(1) Choose a vocabulary

↓

(2) Formalize the scenario

Step 2: Describing the dependencies with first-order formulas

Your vocabulary

$P(u)$:	u is a program library.
$S(u)$:	u is a system library.
$M(u)$:	u is a maintainer.
$U(u)$:	Software package u must be explicitly installed by the user.
$D(u,v)$:	Software package u depends on software package v.
$m(u)$:	Maintainer of software package u

No software package is both a program library and a system library.

```
∀u ¬(P(u) ∧ S(u))
```

System libraries depend only on system libraries.

```
∀u ∀v [(S(u) ∧ D(u,v)) → S(v)]
```

Program libraries may only depend on system libraries which are not maintained by the same maintainer.

```
∀u ∀v [(P(u) ∧ S(v) ∧ D(u,v)) → ¬(m(u) = m(v))]
```

Every software package that must be explicitly installed by the user depends on at least one program library directly.

```
∀u [U(u) → ∃v(P(v) ∧ D(u,v))]
```

Congratulations! You have successfully completed this task.

Iltis: Design ideas

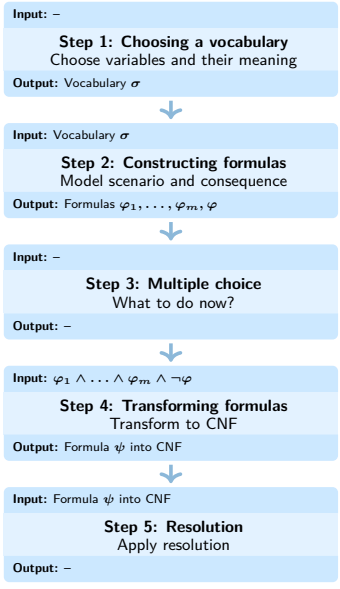
Exercise framework

- Small **educational tasks**, each with inputs and outputs
- Educational tasks flexibly combinable into **workflows**
  - ➡ E.g. reasoning workflows, workflows for testing for equivalence etc.

Iltis: Design ideas

Exercise framework

- Small **educational tasks**, each with inputs and outputs
- Educational tasks flexibly combinable into **workflows**
  - ➡ E.g. reasoning workflows, workflows for testing for equivalence etc.



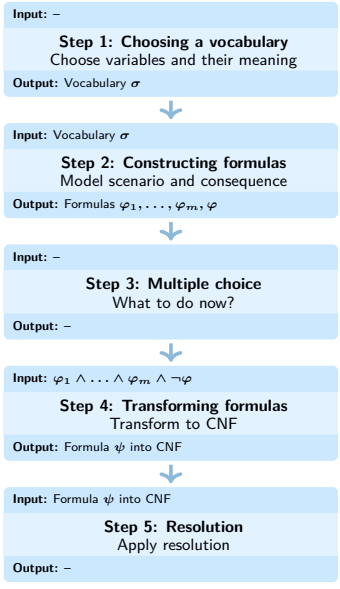
Iltis: Design ideas

Exercise framework

- Small **educational tasks**, each with inputs and outputs
- Educational tasks flexibly combinable into **workflows**
  - ➡ E.g. reasoning workflows, workflows for testing for equivalence etc.

Feedback framework

- **Feedback generators** for chunks of feedback for each educational task
- Generators flexibly combinable into **feedback strategies**
  - ➡ E.g. deep feedback for beginners, less feedback for exam preparation



Iltis: Overview of Educational Tasks – Logic

Task	Propositional logic	Modal logic	First-order logic	CTL
Choosing a vocabulary	✓	✓	✓	✓
Constructing formulas	✓	✓	✓	✓
Transforming formulas	✓	✓	✓	⚙
Demonstrating (un)satisfiability	✓	✓	✓	⚙
Constructing models	✓	✓	✓	⚙
Evaluating formulas	✓	✓	⚙	⚙

✓: supported ⚙: in development

Iltis: Overview of Educational Tasks – TCS

Regular Languages

- Construction of
  - regular expressions
  - deterministic finite automata
  - non-deterministic finite automata
  - regular grammars
- Specifying words
- Specifying Myhill-Nerode classes
- Proving inequality of languages

Context-Free Languages

- Construction of
  - push-down automata
  - deterministic push-down automata
  - context-free grammars
- Specifying words
- Specifying derivations in context-free grammars
- Proving inequality of languages

Computational Complexity

- Understanding algorithmic problems
- Executing computational reductions
- Specifying computational reductions between graph-based problems

General Purpose Tasks

- Multiple choice tasks
- Constructing and manipulating graphs
- Constructing proofs via drag & drop (“proof blocks”)
- Sorting items into buckets (e.g. sorting languages into language classes)

Part II:

Challenges: Theory, Practice, Didactics

Technological support for formal foundations: Theory and practice

A theory challenge: Algorithmically hard problems arise when providing

- feedback and advice for students
- learning analytics to instructors

Technological support for formal foundations: Theory and practice

A theory challenge: Algorithmically hard problems arise when providing

- feedback and advice for students
- learning analytics to instructors

A practical challenge: A large engineering effort is required, e.g., for

- engineering theoretical results towards implementation
- ensuring flexibility, extensibility, usability and maintainability
- ...

Technological support for formal foundations: Theory and practice

A theory challenge: Algorithmically hard problems arise when providing

- feedback and advice for students
- learning analytics to instructors

A practical challenge: A large engineering effort is required, e.g., for

- engineering theoretical results towards implementation
- ensuring flexibility, extensibility, usability and maintainability
- ...

A didactical challenge: Almost no education research on formal foundations of CS

- Collect and publish didactical data
- Identify misconceptions, difficulty generating factors, ...

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Testing correctness (for 99% of grammars)**:**

- Identification of wrong grammars:
  use heuristics
- Identification of correct grammars:
  Solution templates
  + Grammar transformations
  + Isomorphism tests

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Testing correctness (for 99% of grammars)**:**

- Identification of wrong grammars:
  use heuristics
- Identification of correct grammars:
  Solution templates
  + Grammar transformations
  + Isomorphism tests

Pinpointing mistakes

- **Idea:** Restricted context-free languages

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Testing correctness (for 99% of grammars)**:**

- Identification of wrong grammars:
  use heuristics
- Identification of correct grammars:
  Solution templates
  + Grammar transformations
  + Isomorphism tests

Pinpointing mistakes

- **Idea:** Restricted context-free languages
  - A language L is **bounded**, if there are words $w_1, \ldots, w_k$ s.t.

$$L \subseteq w_1^* \ldots w_k^*$$

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Testing correctness (for 99% of grammars)**:**

- Identification of wrong grammars:
  use heuristics
- Identification of correct grammars:
  Solution templates
  + Grammar transformations
  + Isomorphism tests

Pinpointing mistakes

- **Idea:** Restricted context-free languages
  - A language L is **bounded**, if there are words $w_1, \ldots, w_k$ s.t.
    $$L \subseteq w_1^* \ldots w_k^*$$
  - **Ginsburg/Spanier:** Testing equivalence against bounded languages is decidable

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Testing correctness (for 99% of grammars)**:**

- Identification of wrong grammars:
  use heuristics
- Identification of correct grammars:
  Solution templates
  + Grammar transformations
  + Isomorphism tests

Pinpointing mistakes

- **Idea:** Restricted context-free languages
  - A language L is **bounded**, if there are words $w_1, \ldots, w_k$ s.t.
    $$L \subseteq w_1^* \ldots w_k^*$$
  - **Ginsburg/Spanier:** Testing equivalence against bounded languages is decidable
  - **Also:** The w_is and exponents can be identified

Feedback for Context-free Grammar Construction

A typical excercise: *Design a context-free grammar for the language*

$$L = \{a^n b^{n+2} \mid n \in \mathbb{N}\}$$

A feedback strategy:

(1) Correctness:

Your grammar is not correct.

(2) Pinpointing mistakes:

(a) Hint at a mistake

Your grammar describes the language

$$L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

(b) Hint at problems

It might help to have a look at the rule $B \longrightarrow bb$.

Problem: Already testing correctness is undecidable

Testing correctness (for 99% of grammars)**:**

- Identification of wrong grammars:
  use heuristics
- Identification of correct grammars:
  Solution templates
  + Grammar transformations
  + Isomorphism tests

Pinpointing mistakes

- **Idea:** Restricted context-free languages
  - A language L is **bounded**, if there are words $w_1, \ldots, w_k$ s.t.
    $$L \subseteq w_1^* \ldots w_k^*$$
  - **Ginsburg/Spanier:** Testing equivalence against bounded languages is decidable
  - **Also:** The w_is and exponents can be identified
- + a lot of engineering

e.g. to deal with NEXPTIME hardness

Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Challenges

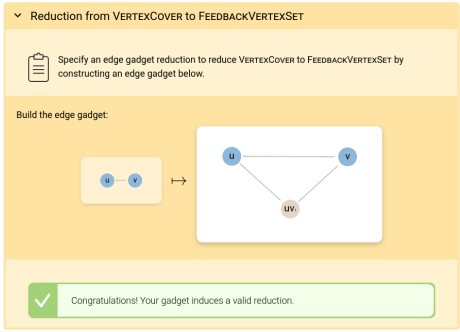
- How to specify reductions?
- How to check correctness and provide feedback?

Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Challenges

- How to specify reductions?
- How to check correctness and provide feedback?



Feedback for Computational Reductions

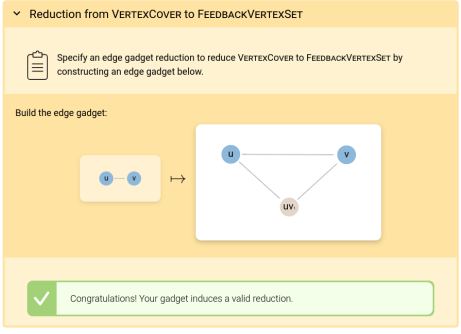
A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Challenges

- How to specify reductions?
- How to check correctness and provide feedback?

Specification language: Cookbook reductions

- Designed for **graphical representation**, **scaffolding**, **algorithmic validation** and **feedback**



Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

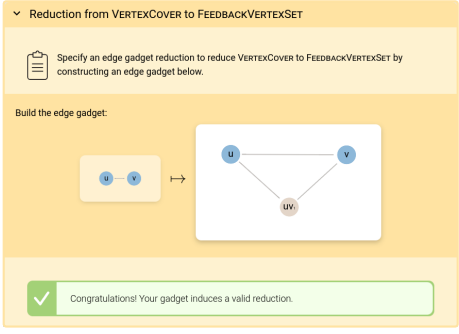
Challenges

- How to specify reductions?
- How to check correctness and provide feedback?

Specification language: Cookbook reductions

- Designed for **graphical representation**, **scaffolding**, **algorithmic validation** and **feedback**
- Allows to specify typical **building blocks**

  ➡ node gadgets, edge gadgets, etc.

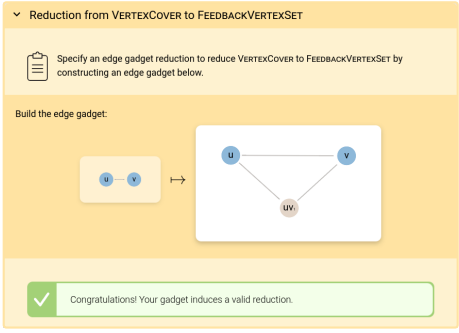


Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Challenges

- How to specify reductions?
- How to check correctness and provide feedback?



Specification language: Cookbook reductions

- Designed for **graphical representation**, **scaffolding**, **algorithmic validation** and **feedback**
- Allows to specify typical **building blocks**

  ➡ node gadgets, edge gadgets, etc.

Correctness and Feedback

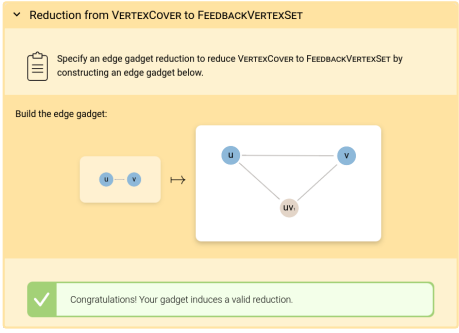
- **Validation:** Is ρ a reduction from P to P'?

Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Challenges

- How to specify reductions?
- How to check correctness and provide feedback?



Specification language: Cookbook reductions

- Designed for **graphical representation**, **scaffolding**, **algorithmic validation** and **feedback**
- Allows to specify typical **building blocks**
  ➡ node gadgets, edge gadgets, etc.

Correctness and Feedback

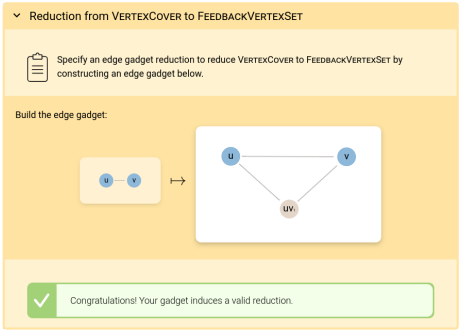
- **Validation:** Is ρ a reduction from P to P'?
- **Theorem:** Validating reductions is undecidable for "classical" reductions, e.g. if ρ is some FO-definable reduction

Feedback for Computational Reductions

A typical excercise: *Design a reduction from VertexCover to FeedbackVertexSet*

Challenges

- How to specify reductions?
- How to check correctness and provide feedback?



Specification language: Cookbook reductions

- Designed for **graphical representation**, **scaffolding**, **algorithmic validation** and **feedback**
- Allows to specify typical **building blocks**
  ➡ node gadgets, edge gadgets, etc.

Correctness and Feedback

- **Validation:** Is ρ a reduction from P to P'?
- **Theorem:** Validating reductions is undecidable for "classical" reductions, e.g. if ρ is some FO-definable reduction
- **Theorem:** Validating reductions is decidable for fixed P, P' and input ρ if
  (a) ρ is some Cookbook reduction, P is arbitrary, and P' is FO-definable
  (b) ρ is some edge gadget reduction, arbitrary P, and P' is MSO-definable

Summary and perspectives

We have seen:

- A web-based educational support system for logic and TCS
- Challenges arising when building such systems

Try the system!

https://iltis.rub.de

Summary and perspectives

We have seen:

- A web-based educational support system for logic and TCS
- Challenges arising when building such systems

Try the system!

https://iltis.rub.de

In my experience:

Interesting research questions arise when building educational support systems for formal foundations

Summary and perspectives

We have seen:

- A web-based educational support system for logic and TCS
- Challenges arising when building such systems

Try the system!

https://iltis.rub.de

In my experience:

Interesting research questions arise when building educational support systems for formal foundations

Perspectives

Research

- Theory: Specification languages, Algorithms, ...
- Engineering: NLP, scalability,...
- CS education: Misconceptions, difficulty generating factors, ...

Summary and perspectives

We have seen:

- A web-based educational support system for logic and TCS
- Challenges arising when building such systems

Try the system!

https://iltis.rub.de

In my experience:

Interesting research questions arise when building educational support systems for formal foundations

Perspectives

Research

- Theory: Specification languages, Algorithms, ...
- Engineering: NLP, scalability,...
- CS education: Misconceptions, difficulty generating factors, ...

Iltis in the wild

- Do you like Iltis?
  - We can provide support for adoption
  - Let us know!

References

Introduction to Iltis

- Marko Schmellenkamp, Fabian Vehlken, and Thomas Zeume. Teaching formal foundations of computer science with Iltis. *Bull. EATCS*, 2024

Logic

- Tristan Kneisel, Fabian Vehlken, and Thomas Zeume. Logical modelling in CS education: Bridging the natural language gap. In *Accepted for AIED 2025*, 2025
- Daniel Neider, Leif Sabellek, Johannes Schmidt, Fabian Vehlken, and Thomas Zeume. Learning tree pattern transformations. In *ICDT 2025*, 2025
- Marko Schmellenkamp, Alexandra Latys, and Thomas Zeume. Discovering and quantifying misconceptions in formal methods using intelligent tutoring systems. In *SIGCSE 2023*, 2023
- Gaetano Geck, Artur Ljulin, Sebastian Peter, Jonas Schmidt, Fabian Vehlken, and Thomas Zeume. Introduction to Iltis: an interactive, web-based system for teaching logic. In *ITiCSE 2018*, 2018

Formal languages

- Marko Schmellenkamp, Thomas Zeume, Sven Argo, Sandra Kiefer, Cedric Siems, and Fynn Stebel. Detecting and explaining (in)equivalence of context-free grammars. 2024
- Marko Schmellenkamp, Dennis Stanglmair, Tilman Michaeli, and Thomas Zeume. Exploring error types in formal languages among students of upper secondary education. In *Koli Calling 2024*, 2024

Reductions

- Julien Grange, Fabian Vehlken, Nils Vortmeier, and Thomas Zeume. Specification and automatic verification of computational reductions. In *MFCS 2024*, 2024
- Tristan Kneisel, Elias Radtke, Marko Schmellenkamp, Fabian Vehlken, and Thomas Zeume. Tool-assisted learning of computational reductions. In *SIGCSE TS 2025*, 2025


Gaetano Geck, Artur Ljulin, Sebastian Peter, Jonas Schmidt, Fabian Vehlken, and Thomas Zeume. Introduction to Iltis: an interactive, web-based system for teaching logic. In *ITiCSE 2018*, 2018.


Julien Grange, Fabian Vehlken, Nils Vortmeier, and Thomas Zeume. Specification and automatic verification of computational reductions. In *MFCS 2024*, 2024.


Tristan Kneisel, Elias Radtke, Marko Schmellenkamp, Fabian Vehlken, and Thomas Zeume. Tool-assisted learning of computational reductions. In *SIGCSE TS 2025*, 2025.


Tristan Kneisel, Fabian Vehlken, and Thomas Zeume. Logical modelling in CS education: Bridging the natural language gap. In *Accepted for AIED 2025*, 2025.


Daniel Neider, Leif Sabellek, Johannes Schmidt, Fabian Vehlken, and Thomas Zeume. Learning tree pattern transformations. In *ICDT 2025*, 2025.


Marko Schmellenkamp, Alexandra Latys, and Thomas Zeume. Discovering and quantifying misconceptions in formal methods using intelligent tutoring systems. In *SIGCSE 2023*, 2023.


Marko Schmellenkamp, Dennis Stanglmair, Tilman Michaeli, and Thomas Zeume. Exploring error types in formal languages among students of upper secondary education. In *Koli Calling 2024*, 2024.


Marko Schmellenkamp, Fabian Vehlken, and Thomas Zeume. Teaching formal foundations of computer science with Iltis. *Bull. EATCS*, 2024.

Marko Schmellenkamp, Thomas Zeume, Sven Argo, Sandra Kiefer, Cedric Siems, and Fynn Stebel. Detecting and explaining (in)equivalence of context-free grammars. 2024.