

A Generic Framework for Compiler Verification in Isabelle/HOL

Martin Desharnais Stefan Brunthaler



Journées Francophones des Langages Applicatifs (JFLA) 2020
samedi 1^{er} février

Isabelle/HOL

Isabelle/HOL is an interactive theorem prover based on higher-order logic (HOL).

Strengths

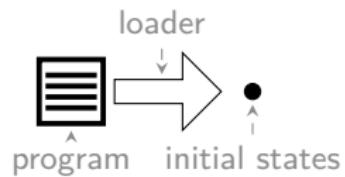
- + small, trustworthy inference kernel
- + Readable proof format
- + Strong proof automation and tooling
 - ▶ general purpose and user defined tactics
 - ▶ integration with automated theorem provers (sledgehammer)
 - ▶ counterexample generators (quickcheck, nitpick)

This talk focuses on locales: a lightweight module system.

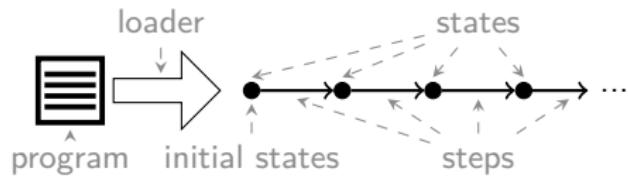
Language semantics



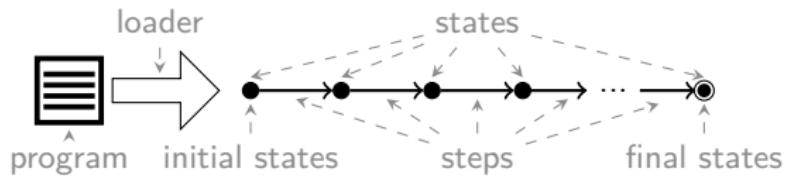
Language semantics



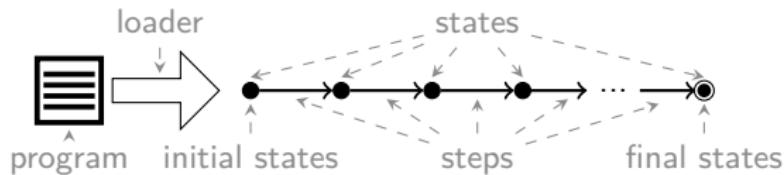
Language semantics



Language semantics



Language semantics

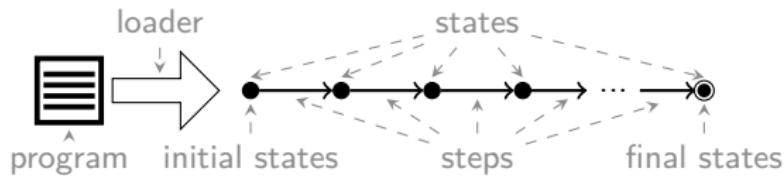


locale semantics =

fixes

step :: *'state* \Rightarrow *'state* \Rightarrow bool **and**
final :: *'state* \Rightarrow bool

Language semantics

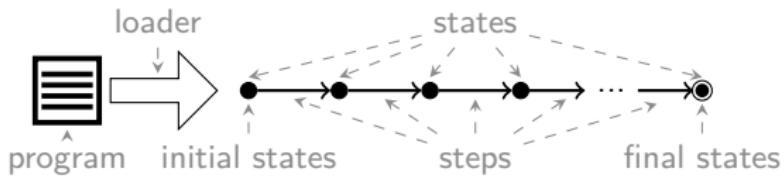


locale semantics =

fixes

step :: *'state* \Rightarrow *'state* \Rightarrow bool **and** $\leftarrow \cdots \cdots$ parameters
final :: *'state* \Rightarrow bool $\leftarrow \cdots \cdots$

Language semantics

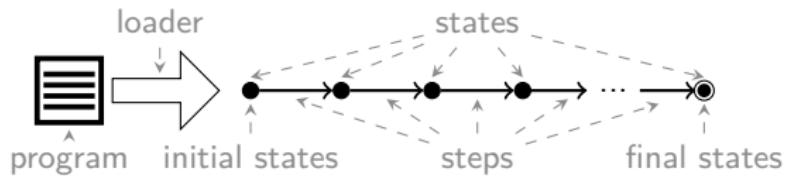


```

    graph TD
        A[locale semantics =] --> B[fixes]
        B --> C["step :: 'state' ⇒ 'state' ⇒ bool and"]
        B --> D["final :: 'state' ⇒ bool"]
        C -.-> E["parameters"]
        D -.-> E
    
```

The diagram illustrates the relationship between locale semantics and its components. It starts with "locale semantics =" at the top left, followed by a dashed arrow pointing down to "fixes". From "fixes", two dashed arrows point to "step :: 'state' ⇒ 'state' ⇒ bool and" and "final :: 'state' ⇒ bool". Finally, from both of these definitions, dashed arrows point to the right, labeled "parameters".

Language semantics



locale semantics =

fixes

type variables

step :: *'state* \Rightarrow *'state* \Rightarrow bool and

final :: *'state* \Rightarrow bool

parameters

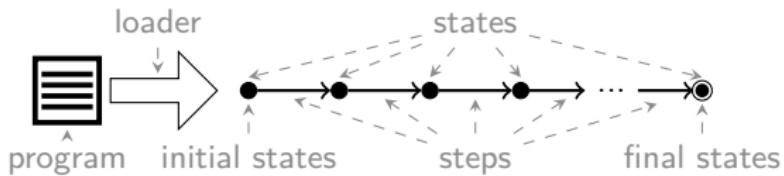
locale language =

semantics \rightarrow \bullet for

\rightarrow and \bullet :: *'state* \Rightarrow bool +

fixes load :: *'prog* \Rightarrow *'state*

Language semantics



locale semantics =

fixes
step :: *'state* \Rightarrow *'state* \Rightarrow bool and
final :: *'state* \Rightarrow bool

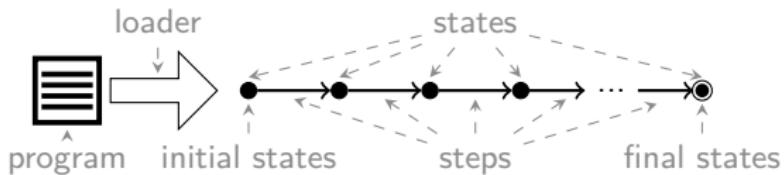
locale language =

semantics \rightarrow \bullet for
 \rightarrow and \bullet :: *'state* \Rightarrow bool +
fixes load :: *'prog* \Rightarrow *'state*

Annotations:

- A dashed arrow points from 'type variables' to the type annotations in the locale semantics code.
- Dashed arrows point from 'parameters' to the parameters in the 'step' and 'final' declarations.
- A dashed arrow points from 'imports' to the 'for' and 'and' declarations in the locale language code.

Language semantics



locale semantics =

fixes

```
step :: 'state ⇒ 'state ⇒ bool and
final :: 'state ⇒ bool
```

type variables

locale language =

semantics → ● for

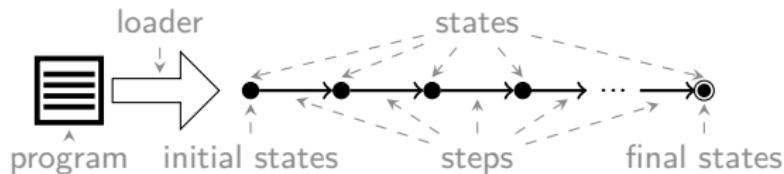
```
→ and ● :: 'state ⇒ bool +
fixes load :: 'prog ⇒ 'state
```

parameters

imports

predicate

Language semantics

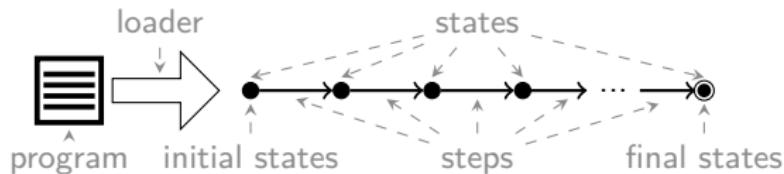


locale semantics =
fixes
step :: *'state* \Rightarrow *'state* \Rightarrow bool and
final :: *'state* \Rightarrow bool

locale language =
semantics \rightarrow \bullet for predicate
 \rightarrow and \bullet :: *'state* \Rightarrow bool + imports
fixes load :: *'prog* \Rightarrow *'state* arbitrary bound variables

Annotations:
"type variables" points to the *'state* in "step :: *'state* \Rightarrow *'state* \Rightarrow bool and".
"parameters" points to the *'state* in "final :: *'state* \Rightarrow bool".
"imports" points to the \rightarrow in "semantics \rightarrow \bullet for predicate".
"arbitrary bound variables" points to the \bullet in "and \bullet :: *'state* \Rightarrow bool +".

Language semantics



locale semantics =
fixes
step :: *'state* \Rightarrow *'state* \Rightarrow bool and
final :: *'state* \Rightarrow bool

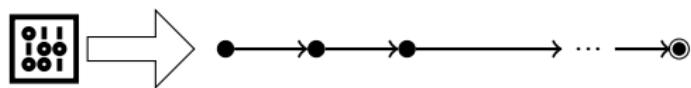
type variables

parameters

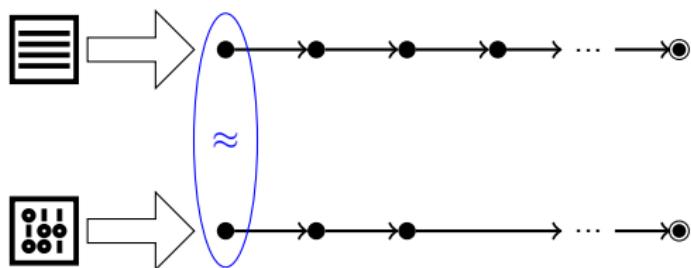
locale language =
semantics \rightarrow \bullet for predicate
 \rightarrow and \bullet :: *'state* \Rightarrow bool +
fixes load :: *'prog* \Rightarrow *'state*

imports
arbitrary bound variables
new parameters

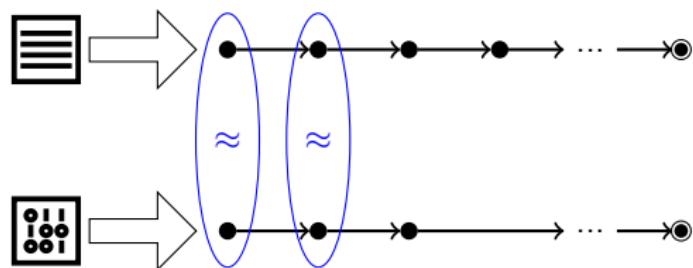
Simulation



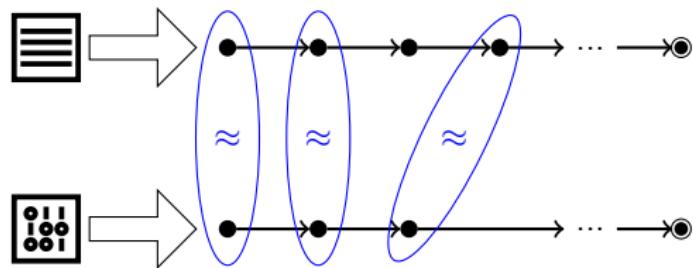
Simulation



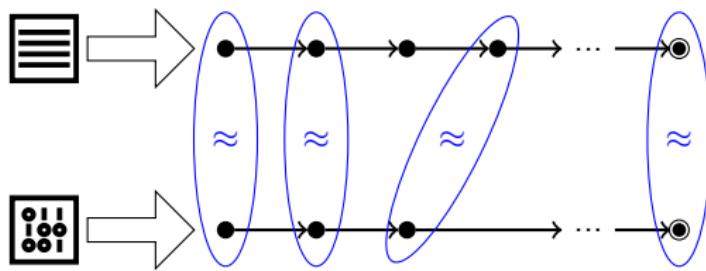
Simulation



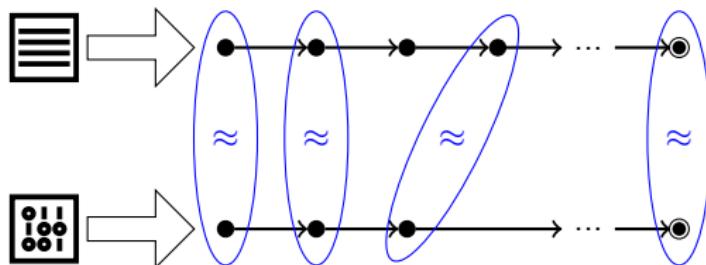
Simulation



Simulation



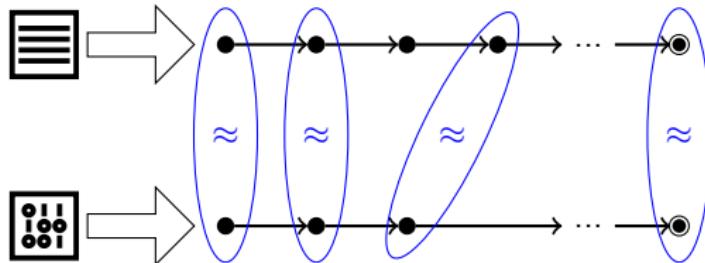
Simulation



```
locale backward-simulation =
L1: semantics →1 ◎1 +
L2: semantics →2 ◎2 +
well-founded □ for
  →1 and ◎1 :: 'state1 ⇒ bool and
  →2 and ◎2 :: 'state2 ⇒ bool and
  □ :: 'index ⇒ 'index ⇒ bool +
fixes match :: 'index ⇒ 'state1 ⇒ 'state2 ⇒ bool
assumes
  match-final: match i s1 s2 ⇒ ◎2 s2 ⇒ ◎1 s1 and
  simulation: match i s1 s2 ⇒ s2 →2 s'2 ⇒
    (exists i' s'1. s1 →1+ s'1 and match i' s'1 s'2 ⇒
```

$$(\exists i' s'_1. s_1 \rightarrow_1^+ s'_1 \wedge \text{match } i' s'_1 s'_2 \Rightarrow \exists i'. \text{match } i' s_1 s'_2 \wedge i' \sqsubset i)$$

Simulation



locale backward-simulation =

L1: semantics $\rightarrow_1 \bullet_1 +$

$\leftarrow \dots$ multiple instances

L2: semantics $\rightarrow_2 \bullet_2 +$

$\leftarrow \dots$

well-founded \sqsubset for

$\rightarrow_1 \text{ and } \bullet_1 :: 'state_1 \Rightarrow \text{bool} \text{ and}$

$\rightarrow_2 \text{ and } \bullet_2 :: 'state_2 \Rightarrow \text{bool} \text{ and}$

$\sqsubset :: 'index \Rightarrow 'index \Rightarrow \text{bool} +$

fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow \text{bool}$

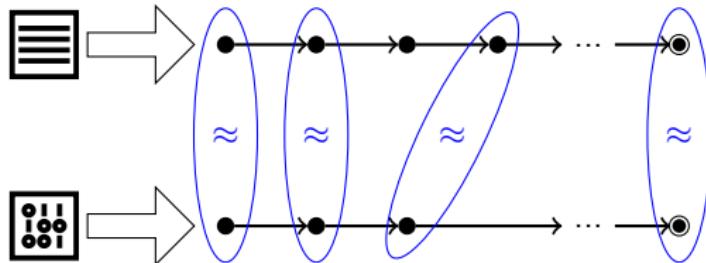
assumes

match-final: match $i s_1 s_2 \implies \bullet_2 s_2 \implies \bullet_1 s_1 \text{ and}$

simulation: match $i s_1 s_2 \implies s_2 \rightarrow_2 s'_2 \implies$

$(\exists i' s'_1. s_1 \rightarrow_1^+ s'_1 \wedge \text{match } i' s'_1 s'_2) \vee (\exists i'. \text{match } i' s_1 s'_2 \wedge i' \sqsubset i)$

Simulation



locale backward-simulation =

L1: semantics $\rightarrow_1 \bullet_1 +$

$\leftarrow \dots$ multiple instances

L2: semantics $\rightarrow_2 \bullet_2 +$

$\leftarrow \dots$

well-founded \sqsubset **for**

$\rightarrow_1 \text{ and } \bullet_1 :: 'state_1 \Rightarrow \text{bool} \text{ and}$

$\rightarrow_2 \text{ and } \bullet_2 :: 'state_2 \Rightarrow \text{bool} \text{ and}$

$\sqsubset :: 'index \Rightarrow 'index \Rightarrow \text{bool} +$

new parameters

fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow \text{bool}$

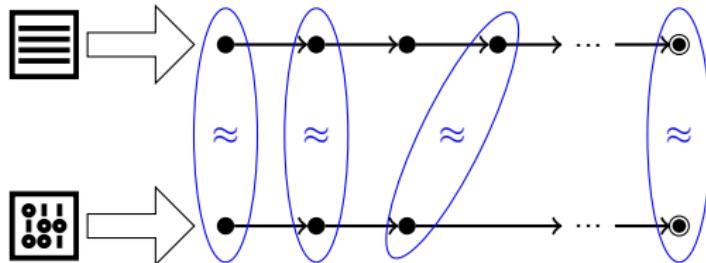
assumes

match-final: match $i s_1 s_2 \implies \bullet_2 s_2 \implies \bullet_1 s_1 \text{ and}$

simulation: match $i s_1 s_2 \implies s_2 \rightarrow_2 s'_2 \implies$

$(\exists i' s'_1. s_1 \rightarrow_1^+ s'_1 \wedge \text{match } i' s'_1 s'_2) \vee (\exists i'. \text{match } i' s_1 s'_2 \wedge i' \sqsubset i)$

Simulation



locale backward-simulation =

L1: semantics $\rightarrow_1 \bullet_1 +$

$\leftarrow \dots$ multiple instances

L2: semantics $\rightarrow_2 \bullet_2 +$

$\leftarrow \dots$

well-founded \square for

$\rightarrow_1 \text{ and } \bullet_1 :: 'state_1 \Rightarrow \text{bool} \text{ and}$

new parameters

$\rightarrow_2 \text{ and } \bullet_2 :: 'state_2 \Rightarrow \text{bool} \text{ and}$

$\square :: 'index \Rightarrow 'index \Rightarrow \text{bool} +$

fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow \text{bool}$

assumptions

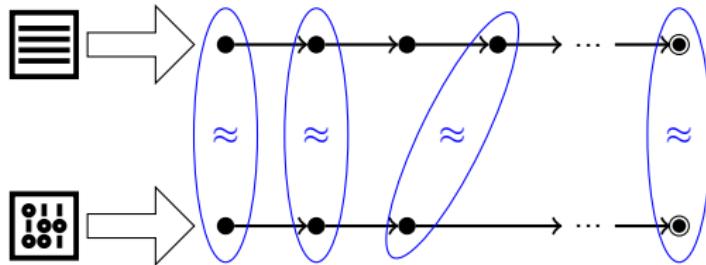
assumes

match-final: match $i s_1 s_2 \implies \bullet_2 s_2 \implies \bullet_1 s_1 \text{ and}$

simulation: match $i s_1 s_2 \implies s_2 \rightarrow_2 s'_2 \implies$

$(\exists i' s'_1. s_1 \rightarrow_1^+ s'_1 \wedge \text{match } i' s'_1 s'_2) \vee (\exists i'. \text{match } i' s_1 s'_2 \wedge i' \sqsubset i)$

Simulation



locale backward-simulation =

L1: semantics $\rightarrow_1 \bullet_1 +$

$\leftarrow \dots$ multiple instances

L2: semantics $\rightarrow_2 \bullet_2 +$

$\leftarrow \dots$

well-founded \square for

$\rightarrow_1 \text{ and } \bullet_1 :: 'state_1 \Rightarrow \text{bool} \text{ and}$

new parameters

$\rightarrow_2 \text{ and } \bullet_2 :: 'state_2 \Rightarrow \text{bool} \text{ and}$

$\square :: 'index \Rightarrow 'index \Rightarrow \text{bool} +$

fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow \text{bool}$

assumptions

assumes

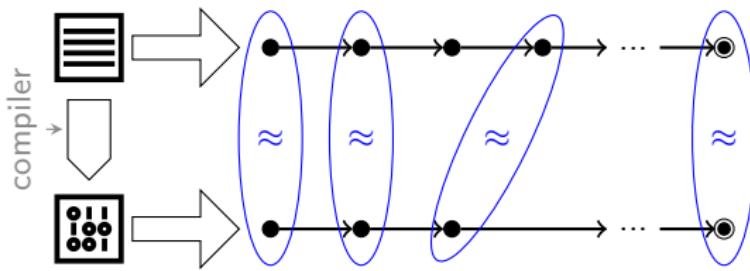
match-final: match $i s_1 s_2 \implies \bullet_2 s_2 \implies \bullet_1 s_1 \text{ and}$

simulation: match $i s_1 s_2 \implies s_2 \rightarrow_2 s'_2 \implies$

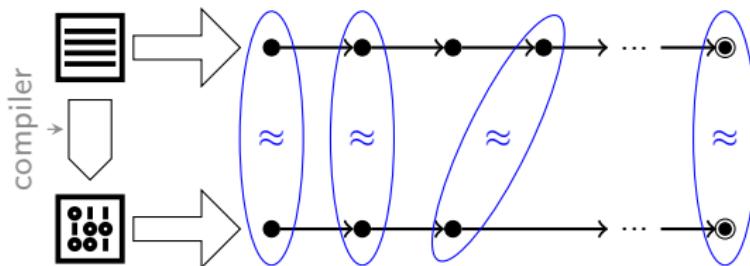
$(\exists i' s'_1. s_1 \rightarrow_1^+ s'_1 \wedge \text{match } i' s'_1 s'_2) \vee (\exists i'. \text{match } i' s_1 s'_2 \wedge i' \sqsubset i)$

transitive closure

Compiler



Compiler



```
locale compiler =
  L1: language →1 ◉1 load1 +
  L2: language →2 ◉2 load2 +
  backward-simulation →1 ◉1 →2 ◉2 ⊓ match for
    →1 and ◉1 and load1 :: 'prog1 ⇒ 'state1 ⇒ bool and
    →2 and ◉2 and load2 :: 'prog2 ⇒ 'state2 ⇒ bool and
    ⊓ and match
  fixes compile :: 'prog1 ⇒ 'prog2 option
  assumes
    compile-load: compile p1 = Some p2 ⇒ ∃i. match i(load1 p1) (load2 p2)
```

Derived lemma (partial correctness)

context compiler **begin**

end

Derived lemma (partial correctness)

context compiler **begin**

$$\frac{\text{compile } p_1 = \text{Some } p_2 \quad \text{load}_2 p_2 \Downarrow b_2 \quad \neg \text{is-wrong } b_2}{\exists b_1 i. \text{load}_1 p_1 \Downarrow b_1 \wedge b_1 \simeq b_2}$$

end

Derived lemma (compiler composition)

compiler $\rightarrow_1 \odot_1 load_1 \rightarrow_2 \odot_2 load_2 \sqsubset_{1-2} match_{1-2} compile_{1-2}$

compiler $\rightarrow_2 \odot_2 load_2 \rightarrow_3 \odot_3 load_3 \sqsubset_{2-3} match_{2-3} compile_{2-3}$

compiler $\rightarrow_1 \odot_1 load_1 \rightarrow_3 \odot_3 load_3 [\dots] [\dots] (compile_{2-3} \Leftarrow compile_{1-2})$

Experience report on locales

Framework

- ▶ Small code base (1 kLOC)
- ▶ Tiny interface (the actual locales)

Instantiations

- ▶ Optimization of virtual machines
- ▶ Three bytecode languages
- ▶ Somewhat bigger (7 kLOC)

Experience report on locales

Framework

- ▶ Small code base (1 kLOC)
- ▶ Tiny interface (the actual locales)

Instantiations

- ▶ Optimization of virtual machines
- ▶ Three bytecode languages
- ▶ Somewhat bigger (7 kLOC)

Pros

- + Separation of concerns (parameters, assumptions, derived results)
- + Multiple abstract data types (e.g. *'state*, *'prog*, etc.)

Experience report on locales

Framework

- ▶ Small code base (1 kLOC)
- ▶ Tiny interface (the actual locales)

Instantiations

- ▶ Optimization of virtual machines
- ▶ Three bytecode languages
- ▶ Somewhat bigger (7 kLOC)

Pros

- + Separation of concerns (parameters, assumptions, derived results)
- + Multiple abstract data types (e.g. *'state*, *'prog*, etc.)

Cons

- Syntactical overhead (predicates, imports, type annotations)
- Only first order type variables (limitation of HOL)

Take away

Locales can be **syntax heavy**, but can lead to **generic frameworks** in software formalizations.

Framework users must prove only **three* lemmata**

- ▶ all final states match a final state
- ▶ there exists a backward simulation
- ▶ compilation and loading lead to matching states

to get

- ▶ partial correctness
- ▶ compositionality

Take away

Locales can be **syntax heavy**, but can lead to **generic frameworks** in software formalizations.

Framework users must prove only **three* lemmata**

- ▶ all final states match a final state
- ▶ there exists a backward simulation
- ▶ compilation and loading lead to matching states

to get

- ▶ partial correctness
- ▶ compositionality

Thank you

Derived results (behaviour)

context semantics **begin**

step [*] $s_1 s_2$	finished step s_2	final s_2
-----------------------------	---------------------	-------------

$s_1 \Downarrow$ Terminates s_2

step [*] $s_1 s_2$	finished step s_2	\neg final s_2
-----------------------------	---------------------	--------------------

$s_1 \Downarrow$ Goes-wrong s_2

step ^{∞} s_1

$s_1 \Downarrow$ Diverges

end