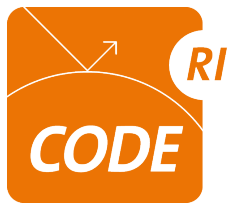
A Generic Framework for Compiler Verification in Isabelle/HOL

Martin Desharnais Stefan Brunthaler



Research Institute
Cyber Defence
Universität der Bundeswehr München

Journées Francophones des Langages Applicatifs (JFLA) 2020

samedi 1er février

Isabelle/HOL

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

Strengths

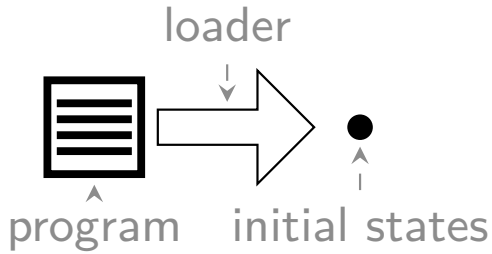
+ small, thrustworthy inference kernel
+ Readable proof format
+ Strong proof automation and tooling
  - general purpose and user defined tactics
  - integration with automated theorem provers (sledgehammer)
  - counterexample generators (quickcheck, nitpick)

This talk focuses on locales: a lightweight module system.

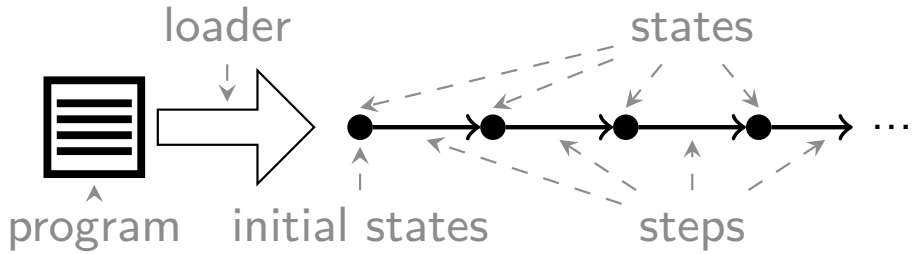
Language semantics

program

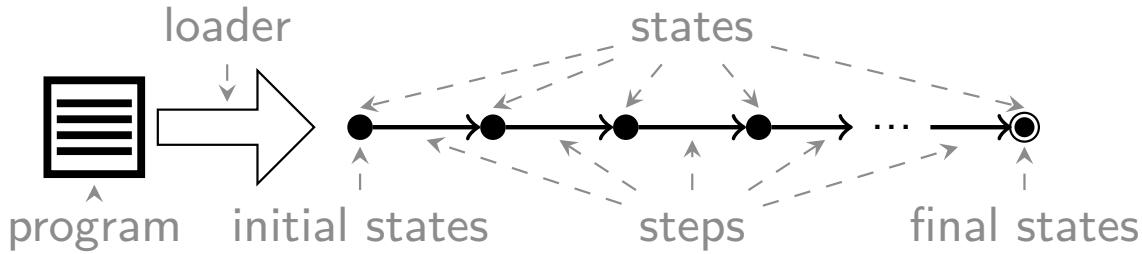
Language semantics



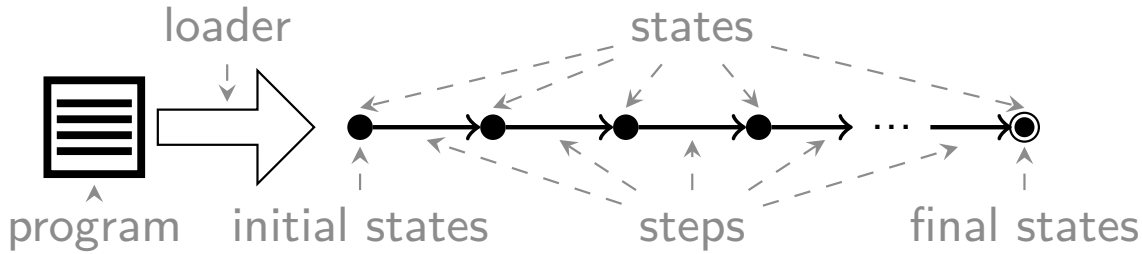
Language semantics



Language semantics

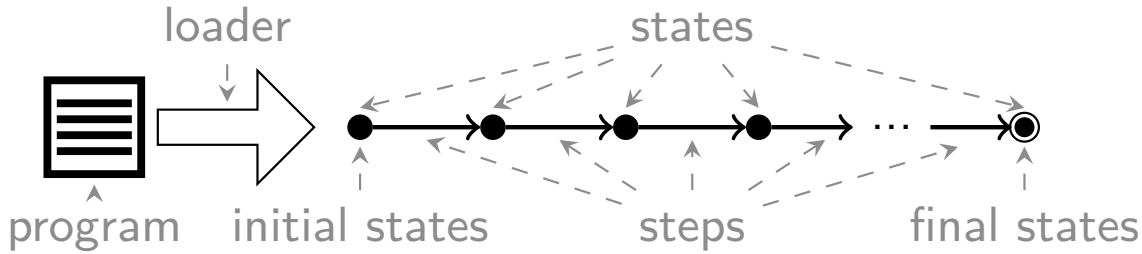


Language semantics



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

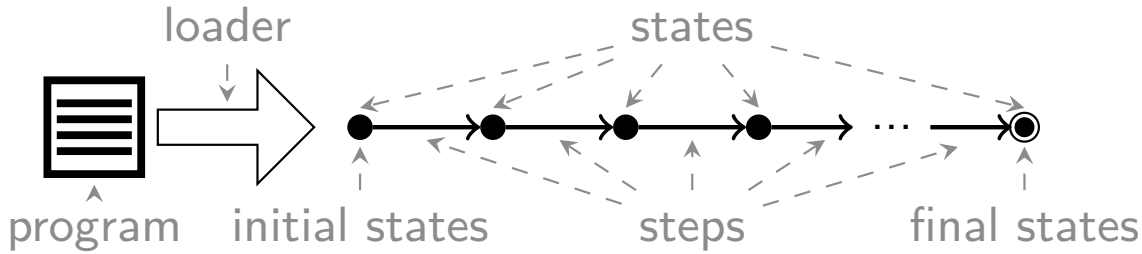
Language semantics



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

parameters

Language semantics

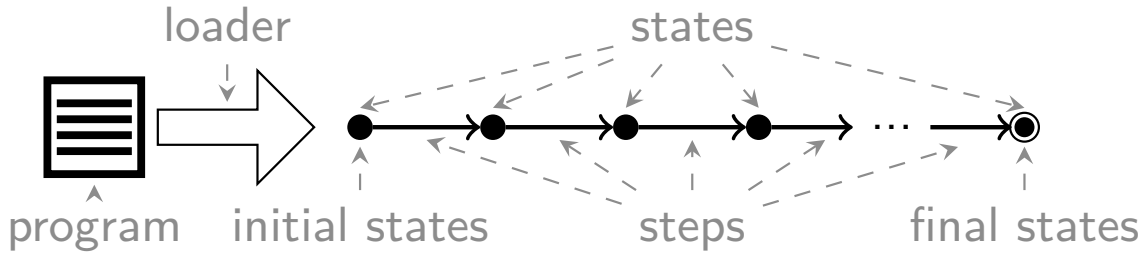


type variables

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

parameters

Language semantics



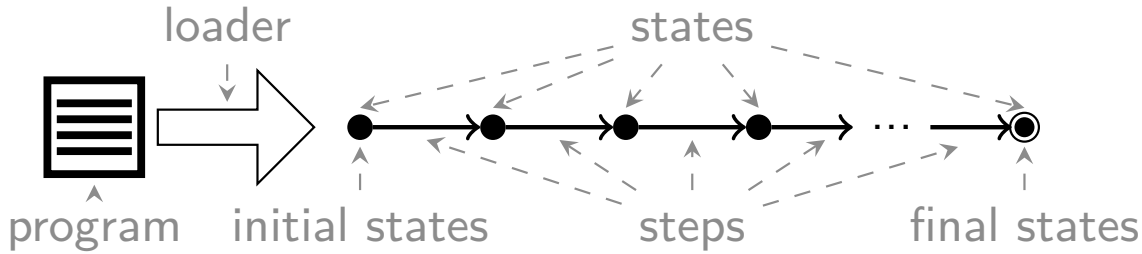
type variables

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

parameters

locale language =
  semantics → ◉ **for**
    → **and** ◉ :: *'state* ⇒ bool +
  fixes load :: *'prog* ⇒ *'state*

Language semantics



type variables

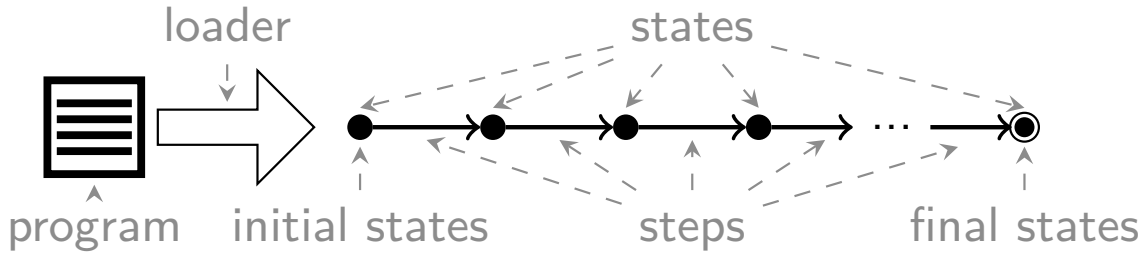
locale semantics =
  fixes
    step :: *'state* ⇒ *'state* ⇒ bool **and**
    final :: *'state* ⇒ bool

parameters

locale language =
  semantics → ◉ **for**
    → **and** ◉ :: *'state* ⇒ bool +
  fixes load :: *'prog* ⇒ *'state*

imports

Language semantics



type variables

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

parameters

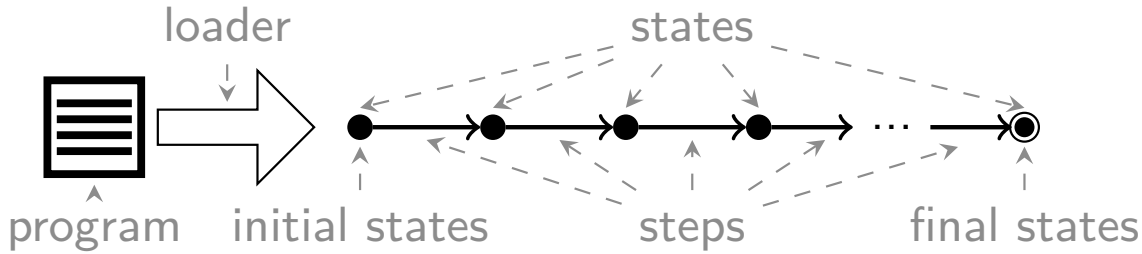
predicate

locale language =
  semantics ⟶ ◉ **for**

imports

    ⟶ **and** ◉ :: *'state* ⇒ bool +
  fixes load :: *'prog* ⇒ *'state*

Language semantics



type variables

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

parameters

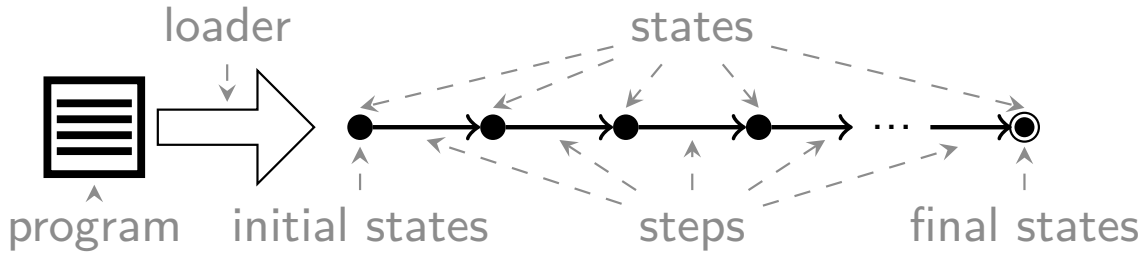
locale language =
  semantics ⟶ ◉ **for**
    ⟶ **and** ◉ :: *'state* ⇒ bool +
  fixes load :: *'prog* ⇒ *'state*

predicate

imports

arbitrary bound variables

Language semantics



type variables

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

parameters

predicate

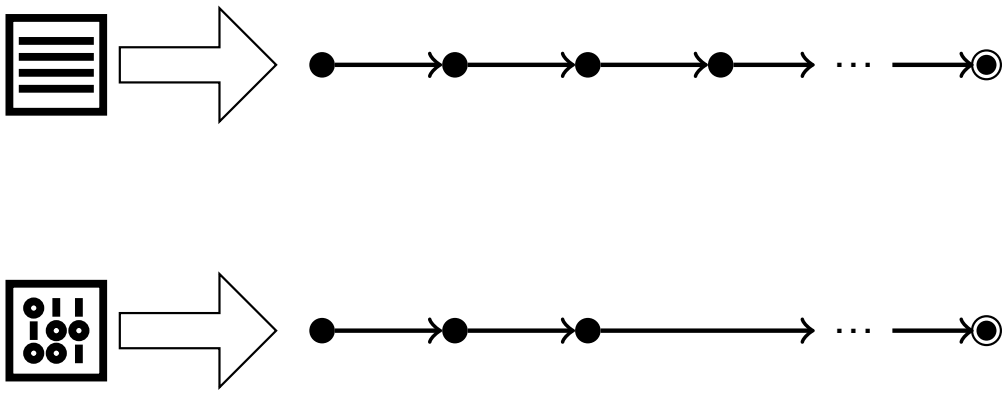
locale language =
  semantics ⟶ ◉ **for**
    ⟶ **and** ◉ :: *'state* ⇒ bool +
  fixes load :: *'prog* ⇒ *'state*

imports

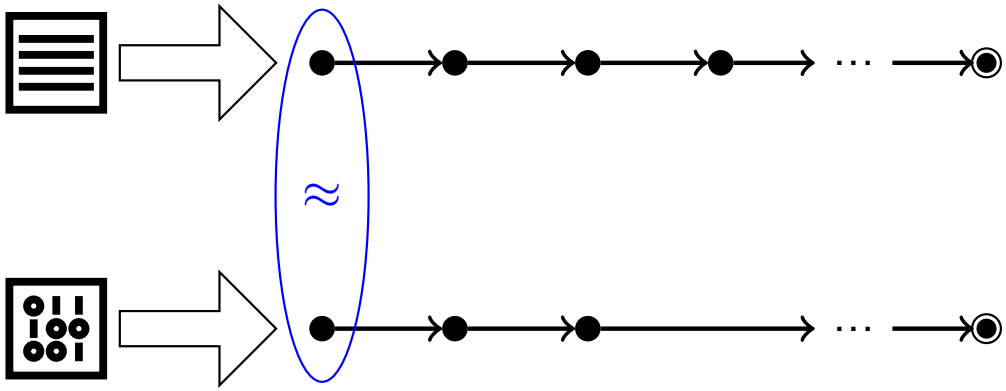
arbitrary bound variables

new parameters

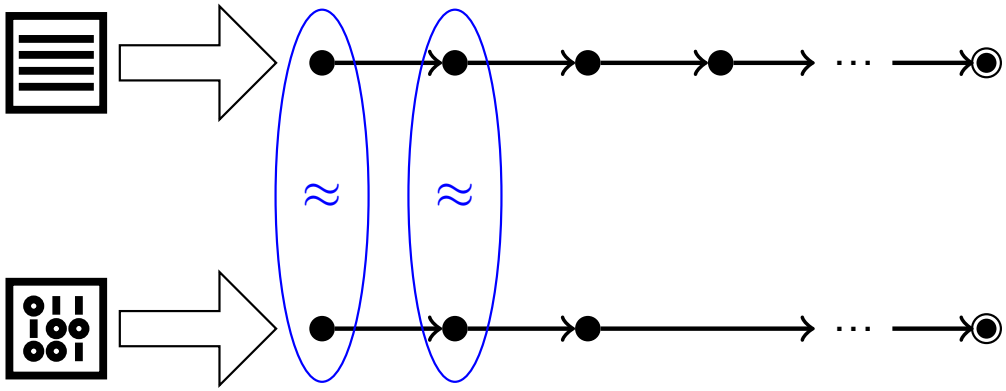
Simulation



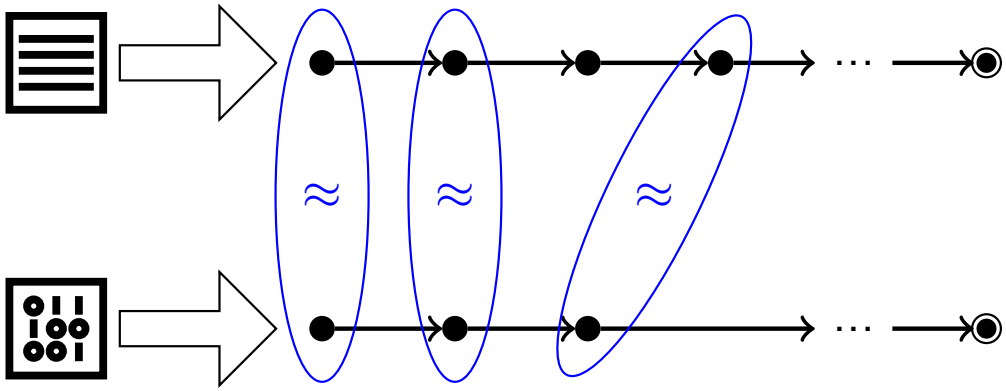
Simulation



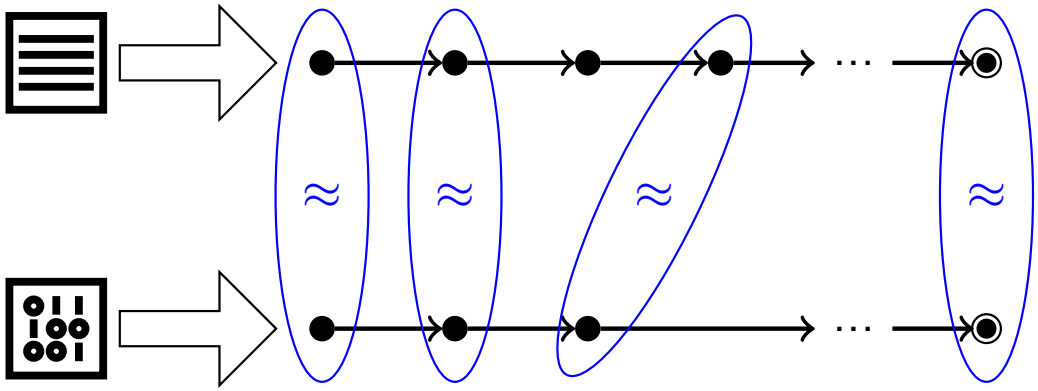
Simulation



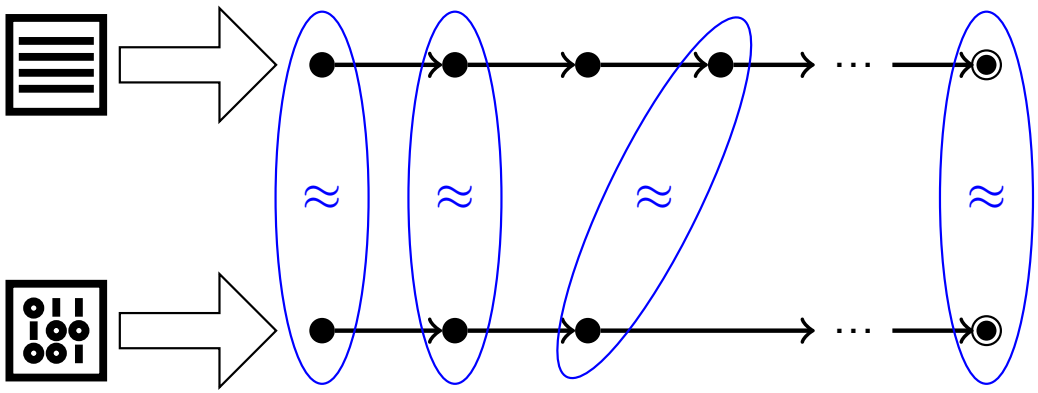
Simulation



Simulation

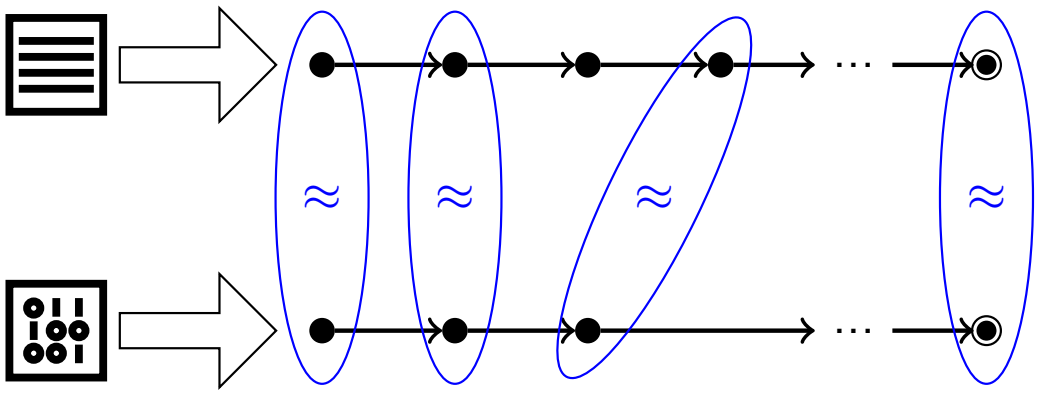


Simulation



locale backward-simulation =
  L1: semantics $\rightarrow_1$ $\bullet_1$ +
  L2: semantics $\rightarrow_2$ $\bullet_2$ +
  well-founded $\sqsubset$ **for**
    $\rightarrow_1$ **and** $\bullet_1$ :: $'state_1 \Rightarrow$ bool **and**
    $\rightarrow_2$ **and** $\bullet_2$ :: $'state_2 \Rightarrow$ bool **and**
    $\sqsubset$ :: $'index \Rightarrow 'index \Rightarrow$ bool +
  fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow$ bool
  assumes
    match-final: match $i\ s_1\ s_2 \Longrightarrow \bullet_2\ s_2 \Longrightarrow \bullet_1\ s_1$ **and**
    simulation: match $i\ s_1\ s_2 \Longrightarrow s_2 \rightarrow_2 s_2' \Longrightarrow$
      $(\exists i'\ s_1'.\ s_1 \rightarrow_1^+ s_1' \wedge \text{match}\ i'\ s_1'\ s_2') \vee (\exists i'.\ \text{match}\ i'\ s_1\ s_2' \wedge i' \sqsubset i)$

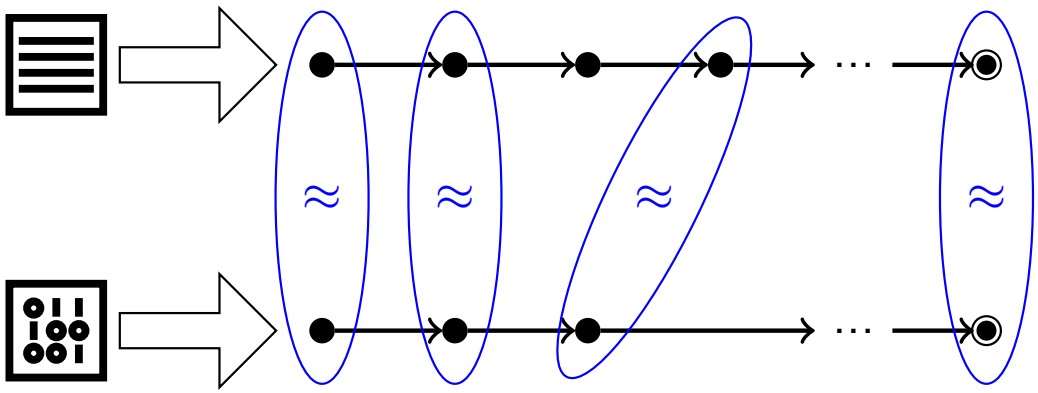
Simulation



locale backward-simulation =
  L1: semantics $\rightarrow_1$ $\circledcirc_1$ +
  L2: semantics $\rightarrow_2$ $\circledcirc_2$ +
  well-founded $\sqsubset$ **for**
    $\rightarrow_1$ **and** $\circledcirc_1$ :: $'state_1 \Rightarrow$ bool **and**
    $\rightarrow_2$ **and** $\circledcirc_2$ :: $'state_2 \Rightarrow$ bool **and**
    $\sqsubset$ :: $'index \Rightarrow 'index \Rightarrow$ bool +
  fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow$ bool
  assumes
    match-final: match $i\ s_1\ s_2 \Longrightarrow \circledcirc_2\ s_2 \Longrightarrow \circledcirc_1\ s_1$ **and**
    simulation: match $i\ s_1\ s_2 \Longrightarrow s_2 \rightarrow_2 s_2' \Longrightarrow$
      $(\exists i'\ s_1'.\ s_1 \rightarrow_1^+ s_1' \wedge \text{match}\ i'\ s_1'\ s_2') \vee (\exists i'.\ \text{match}\ i'\ s_1\ s_2' \wedge i' \sqsubset i)$

multiple instances

Simulation

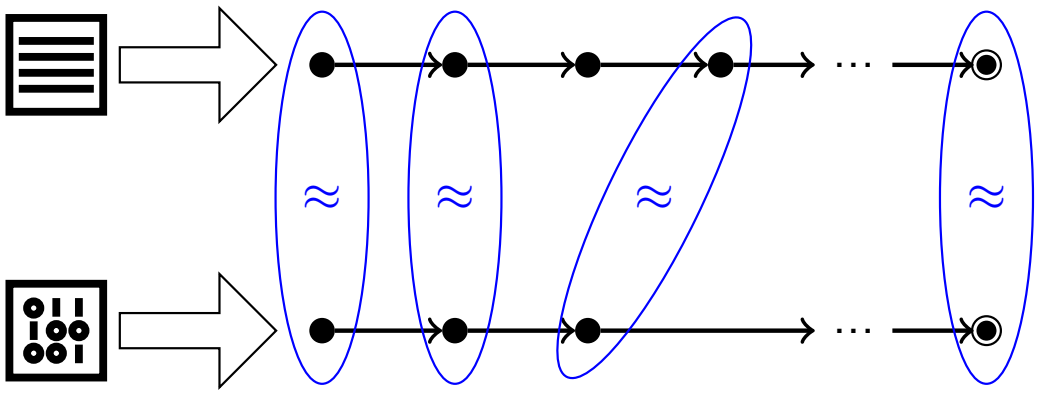


locale backward-simulation =
  L1: semantics $\rightarrow_1$ $\bullet_1$ +
  L2: semantics $\rightarrow_2$ $\bullet_2$ +
  well-founded $\sqsubset$ **for**
    $\rightarrow_1$ **and** $\bullet_1$ :: $'state_1 \Rightarrow$ bool **and**
    $\rightarrow_2$ **and** $\bullet_2$ :: $'state_2 \Rightarrow$ bool **and**
    $\sqsubset$ :: $'index \Rightarrow 'index \Rightarrow$ bool +
  fixes match :: $'index \Rightarrow 'state_1 \Rightarrow 'state_2 \Rightarrow$ bool
  assumes
    match-final: match $i\ s_1\ s_2 \Longrightarrow \bullet_2\ s_2 \Longrightarrow \bullet_1\ s_1$ **and**
    simulation: match $i\ s_1\ s_2 \Longrightarrow s_2 \rightarrow_2 s_2' \Longrightarrow$
      $(\exists i'\ s_1'.\ s_1 \rightarrow_1^+ s_1' \wedge \text{match}\ i'\ s_1'\ s_2') \vee (\exists i'.\ \text{match}\ i'\ s_1\ s_2' \wedge i' \sqsubset i)$

multiple instances

new parameters

Simulation



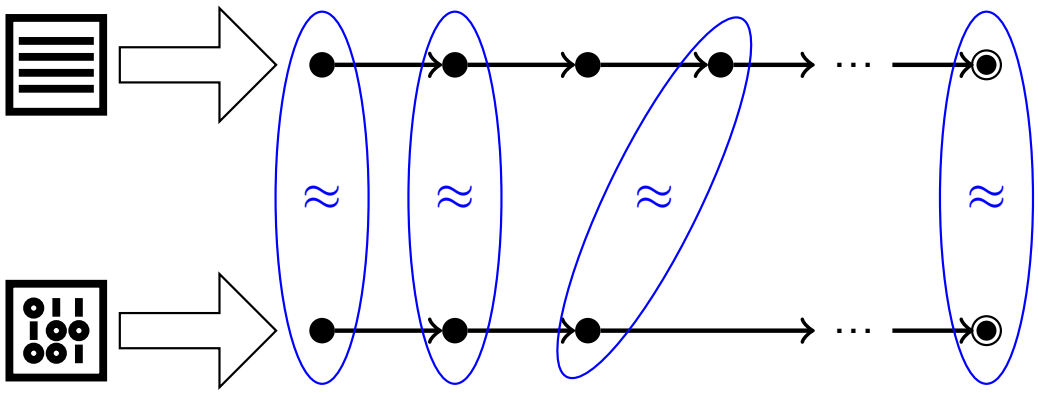
locale backward-simulation =
  L1: semantics $\rightarrow_1$ $\bullet_1$ +
  L2: semantics $\rightarrow_2$ $\bullet_2$ +
  well-founded $\sqsubset$ **for**
    $\rightarrow_1$ **and** $\bullet_1$ :: *'state*$_1$ $\Rightarrow$ bool **and**
    $\rightarrow_2$ **and** $\bullet_2$ :: *'state*$_2$ $\Rightarrow$ bool **and**
    $\sqsubset$ :: *'index* $\Rightarrow$ *'index* $\Rightarrow$ bool +
  fixes match :: *'index* $\Rightarrow$ *'state*$_1$ $\Rightarrow$ *'state*$_2$ $\Rightarrow$ bool
  assumes
    match-final: match $i\ s_1\ s_2 \implies \bullet_2\ s_2 \implies \bullet_1\ s_1$ **and**
    simulation: match $i\ s_1\ s_2 \implies s_2 \rightarrow_2 s_2' \implies$
      $(\exists i'\ s_1'.\ s_1 \rightarrow_1^+ s_1' \wedge \text{match}\ i'\ s_1'\ s_2') \vee (\exists i'.\ \text{match}\ i'\ s_1\ s_2' \wedge i' \sqsubset i)$

multiple instances

new parameters

assumptions

Simulation



locale backward-simulation =

L1: semantics $\rightarrow_1$ $\bullet_1$ +

L2: semantics $\rightarrow_2$ $\bullet_2$ +

multiple instances

well-founded $\sqsubset$ **for**

$\rightarrow_1$ **and** $\bullet_1$ :: *'state*$_1$ $\Rightarrow$ bool **and**

$\rightarrow_2$ **and** $\bullet_2$ :: *'state*$_2$ $\Rightarrow$ bool **and**

$\sqsubset$ :: *'index* $\Rightarrow$ *'index* $\Rightarrow$ bool +

new parameters

fixes match :: *'index* $\Rightarrow$ *'state*$_1$ $\Rightarrow$ *'state*$_2$ $\Rightarrow$ bool

assumptions

assumes

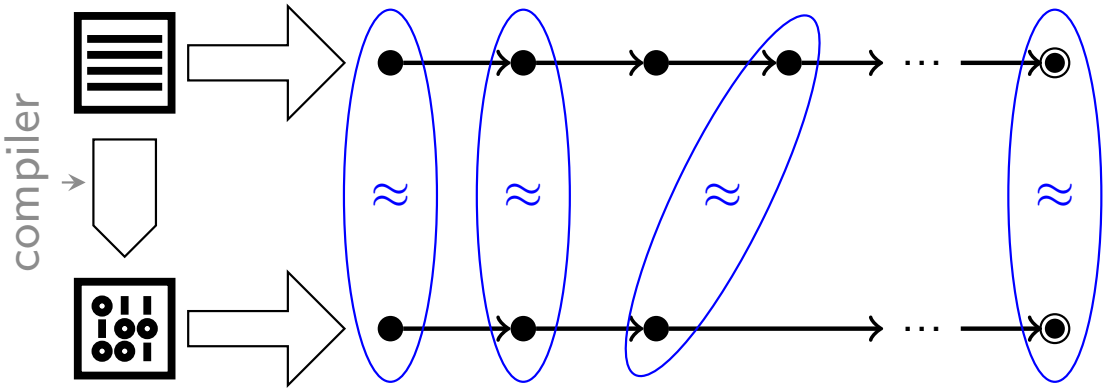
match-final: match $i\ s_1\ s_2 \Longrightarrow \bullet_2\ s_2 \Longrightarrow \bullet_1\ s_1$ **and**

simulation: match $i\ s_1\ s_2 \Longrightarrow s_2 \rightarrow_2 s_2' \Longrightarrow$

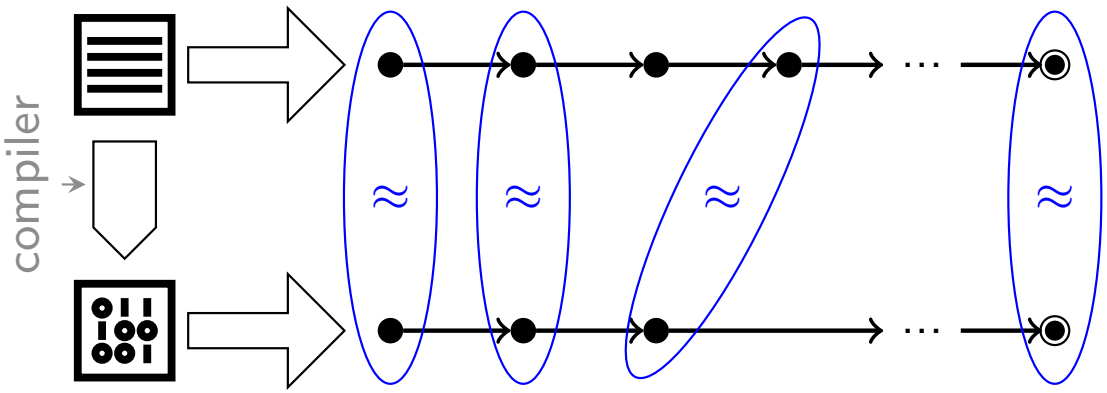
$(\exists i'\ s_1'.\ s_1 \rightarrow_1^+ s_1' \wedge \text{match}\ i'\ s_1'\ s_2') \vee (\exists i'.\ \text{match}\ i'\ s_1\ s_2' \wedge i' \sqsubset i)$

transitive closure

Compiler



Compiler



locale compiler =
  L1: language $\rightarrow_1$ $\bullet_1$ load$_1$ +
  L2: language $\rightarrow_2$ $\bullet_2$ load$_2$ +
  backward-simulation $\rightarrow_1$ $\bullet_1$ $\rightarrow_2$ $\bullet_2$ $\sqsubset$ match **for**
    $\rightarrow_1$ **and** $\bullet_1$ **and** load$_1$ :: *'prog*$_1$ $\Rightarrow$ *'state*$_1$ $\Rightarrow$ bool **and**
    $\rightarrow_2$ **and** $\bullet_2$ **and** load$_2$ :: *'prog*$_2$ $\Rightarrow$ *'state*$_2$ $\Rightarrow$ bool **and**
    $\sqsubset$ **and** match
  fixes compile :: *'prog*$_1$ $\Rightarrow$ *'prog*$_2$ option
  assumes
    compile-load: compile p_1 = Some p_2 $\Longrightarrow$ $\exists i$. match i (load$_1$ p_1) (load$_2$ p_2)

Derived lemma (partial correctness)

context compiler **begin**

end

Derived lemma (partial correctness)

context compiler **begin**

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

end

Derived lemma (compiler composition)

$$\frac{\mathsf{compiler} \rightarrow_1 \circledcirc_1 \mathit{load}_1 \rightarrow_2 \circledcirc_2 \mathit{load}_2 \sqsubset_{1\text{-}2} \mathit{match}_{1\text{-}2}\ \mathit{compile}_{1\text{-}2} \qquad \mathsf{compiler} \rightarrow_2 \circledcirc_2 \mathit{load}_2 \rightarrow_3 \circledcirc_3 \mathit{load}_3 \sqsubset_{2\text{-}3} \mathit{match}_{2\text{-}3}\ \mathit{compile}_{2\text{-}3}}{\mathsf{compiler} \rightarrow_1 \circledcirc_1 \mathit{load}_1 \rightarrow_3 \circledcirc_3 \mathit{load}_3\ [\ldots]\ [\ldots]\ (\mathit{compile}_{2\text{-}3} \Leftarrow \mathit{compile}_{1\text{-}2})}$$

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**

$$\frac{\mathsf{step}^*\ s_1\ s_2 \qquad \mathsf{finished\ step}\ s_2 \qquad \mathsf{final}\ s_2}{s_1 \Downarrow \mathsf{Terminates}\ s_2}$$

$$\frac{\mathsf{step}^*\ s_1\ s_2 \qquad \mathsf{finished\ step}\ s_2 \qquad \neg\,\mathsf{final}\ s_2}{s_1 \Downarrow \mathsf{Goes\text{-}wrong}\ s_2}$$

$$\frac{\mathsf{step}^\infty\ s_1}{s_1 \Downarrow \mathsf{Diverges}}$$

end