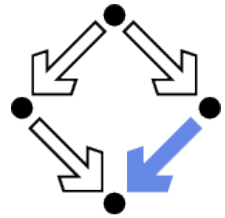
The RISC ProgramExplorer

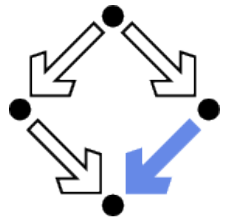
Reasoning about Programs as State Relations

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

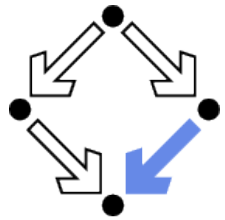
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.jku.at



Program Verification (Classical)



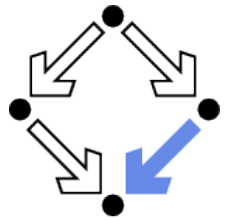
Program Verification (Classical)



P

- Program P

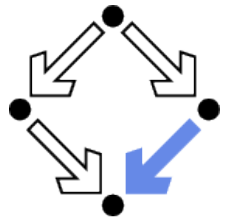
Program Verification (Classical)



$$P \quad + \quad S$$

- Program P
- Specification S

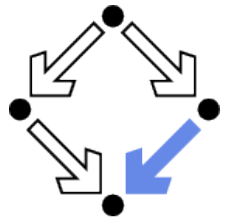
Program Verification (Classical)



$$P^A \quad + \quad S$$

- Program P
- Specification S
- Annotation A

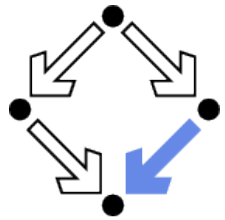
Program Verification (Classical)



$$P^A \quad + \quad S \quad \begin{array}{c} \nearrow \\ \longrightarrow \\ \searrow \end{array} \quad \begin{array}{c} VC_1 \\ \vdots \\ VC_i \\ \vdots \\ VC_n \end{array}$$

- Program P
- Specification S
- Annotation A
- Verification Conditions VC_i

Program Verification (Classical)

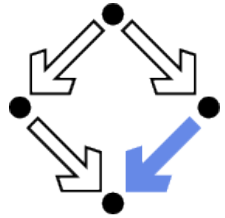


$$P^A \quad + \quad S \quad \nearrow \; \vdots \; VC_1$$
$$\qquad\qquad\quad \longrightarrow \quad VC_i$$
$$\qquad\qquad\quad \searrow \; \vdots \; VC_n$$

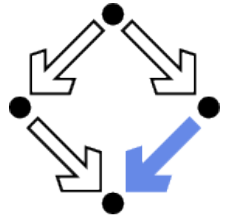
- Program P
- Specification S
- Annotation A
- Verification Conditions VC_i

Problem: only when proving VC_i, we learn whether P, A, S "match".

Soundness of Verification

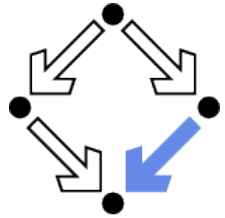


Soundness of Verification



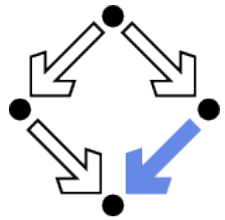
- Program P . . . command

Soundness of Verification



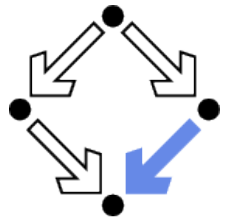
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$

Soundness of Verification



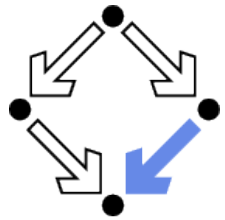
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula

Soundness of Verification



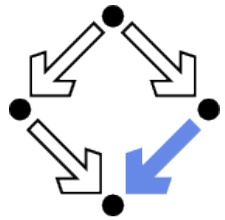
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$

Soundness of Verification



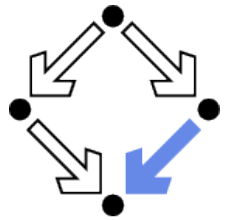
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$
- Verification Condition VC_i . . . formula

Soundness of Verification



- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$
- Verification Condition VC_i . . . formula
  - Condition semantics $[\![VC_i]\!] \in \{\text{true}, \text{false}\}$

Soundness of Verification

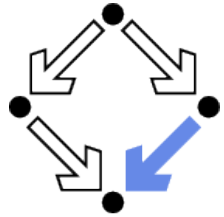


- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$
- Verification Condition VC_i . . . formula
  - Condition semantics $[\![VC_i]\!] \in \{\text{true}, \text{false}\}$

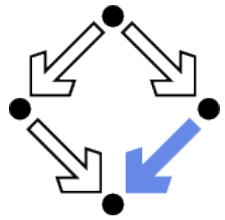
$$[\![VC_i]\!] \Rightarrow \forall s, s' \in State : [\![P]\!](s, s') \Rightarrow [\![S]\!](s, s')$$

If the verification conditions are valid, the state transitions performed by the program are allowed by the specification.

Program Reasoning (Alternative)



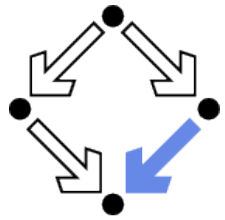
Program Reasoning (Alternative)



P^A

- Annotated program P^A

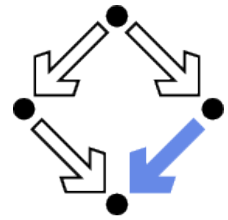
Program Reasoning (Alternative)



$$P^A \longrightarrow F$$

- Annotated program P^A
- Transition formula F

Program Reasoning (Alternative)

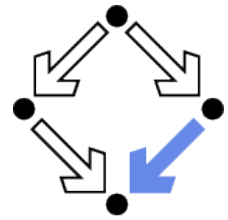


$$P^A \longrightarrow \circlearrowleft F$$

- Annotated program P^A
- Transition formula F

F represents the "semantic essence" of P^A that is open for investigation.

Program Reasoning (Alternative)

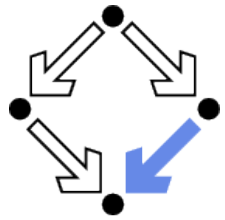


$$P^A \longrightarrow \circlearrowleft F \quad + \quad S$$

- Annotated program P^A
- Transition formula F
- Specification S

F represents the "semantic essence" of P^A that is open for investigation.

Program Reasoning (Alternative)

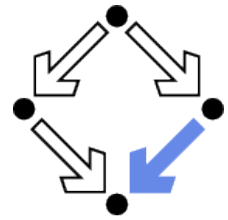


$$P^A \longrightarrow \overset{\circlearrowleft}{F} \quad + \quad S \longrightarrow F \Rightarrow S$$

- Annotated program P^A
- Transition formula F
- Specification S
- Verification Condition $F \Rightarrow S$

F represents the "semantic essence" of P^A that is open for investigation.

Program Reasoning (Alternative)



$$P^A \longrightarrow \circlearrowleft F \quad + \quad S \longrightarrow F \Rightarrow S$$

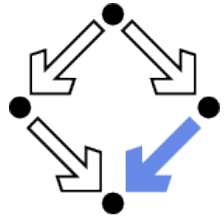
$$\searrow$$

$$TC_1$$
$$\ldots$$
$$TC_n$$

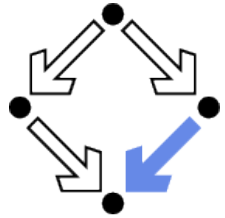
- Annotated program P^A
- Transition formula F
- Specification S
- Verification Condition $F \Rightarrow S$
- Translation Conditions $TC_1, \ldots, TC_n$

F represents the "semantic essence" of P^A that is open for investigation.

Soundness of Translation

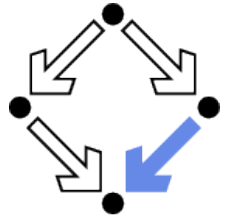


Soundness of Translation



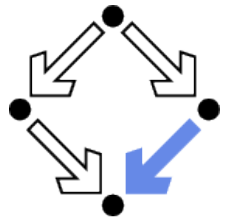
- Program P . . . command

Soundness of Translation



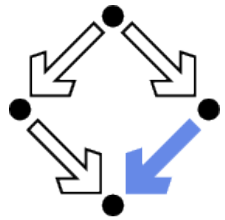
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$

Soundness of Translation



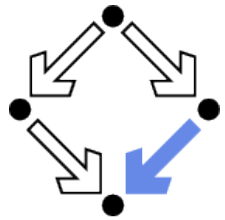
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula

Soundness of Translation



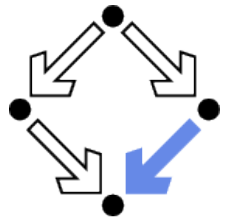
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$

Soundness of Translation



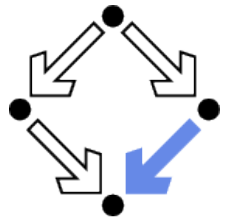
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$
- Translation Conditions TC_i . . . formula

Soundness of Translation



- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$
- Translation Conditions TC_i . . . formula
  - Condition semantics $[\![TC_i]\!] \in \{\text{true}, \text{false}\}$

Soundness of Translation

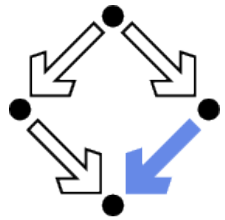


- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$
- Translation Conditions TC_i . . . formula
  - Condition semantics $[\![TC_i]\!] \in \{\text{true}, \text{false}\}$

$$[\![TC_i]\!] \Rightarrow \forall s, s' \in State : [\![P]\!](s, s') \Rightarrow [\![F]\!](s, s')$$

If the translation conditions are valid, the state transitions performed by the program are captured by the transition formula.

Translation Calculus (Excerpt)

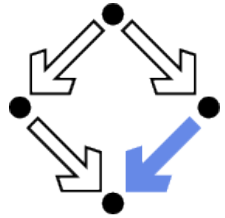


$$\frac{e \simeq_h t}{x = e : [\text{var } x = t]^{\{x\}}_{\text{true},h}} \qquad \frac{c : [f]^{xs}_{g,h} \qquad x \notin xs}{c : [f \wedge \text{var } x = \text{old } x]^{xs \cup \{x\}}_{g,h}}$$

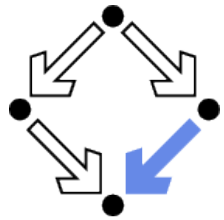
$$\frac{c_1 : [f_1]^{xs}_{g_1,h_1} \qquad c_2 : [f_2]^{xs}_{g_2,h_2} \qquad \text{pre}(c_2, h_2) = h_3}{\{c_1 ; c_2\} : [\exists ys : f_1[ys/\text{var } xs] \wedge f_2[ys/\text{old } xs]]^{xs}_{g_1 \wedge g_2, h_1 \wedge h_3}}$$

Analogous calculus for translation of command to termination condition.

Example



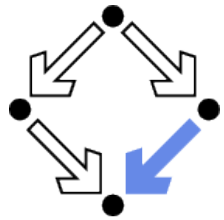
Example



```
if (n < 0)
  s = -1;
else {
  var i;
  s = 0;
  i = 1;
  while (i <= n) {
    s = s+i;
    i = i+1;
  }
}
```

F_1, F_2, F_3, F_s, F_4, F_5, F_b, $F^{,T}$, F_w, F_v, F_e, F_c

Example

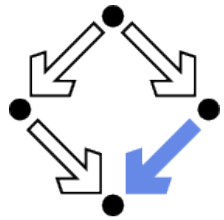


```
if (n < 0)
  s = -1;                F1
else {
  var i;
   s = 0;                F2  } Fs
   i = 1;                F3  }
   while (i <= n) {
     s = s+i;            F4  } Fb  } Fw  } Fv  } Fe  } Fc
     i = i+1;            F5  }
   }^{F,T}
}
```

$F :\Leftrightarrow 1 <= \text{var } i <= \text{var } n{+}1 \text{ and var } s = \sum_{j=1}^{\text{var } i\text{-}1} j$

$T := \text{var } n \text{ - var } i + 1$

Example



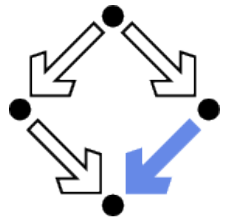
```
if (n < 0)
  s = -1;                F1
else {
  var i;
   s = 0;                F2 } Fs
   i = 1;                F3 }
   while (i <= n) {
     s = s+i;            F4 } Fb } Fw } Fv } Fe } Fc
     i = i+1;            F5 }
   }^{F,T}
}
```

$F :\Leftrightarrow 1 <= \text{var } i <= \text{var } n{+}1 \text{ and var } s = \sum_{j=1}^{\text{var } i\text{-}1} j$

$T := \text{var } n \text{ - var } i + 1$

$$F_c \quad \Leftrightarrow \quad [\text{if old } n < 0 \text{ then var } s = \text{-}1 \text{ else var } s = \sum_{j=1}^{\text{old } n} j]^{\{s\}}$$

Example



```
if (n < 0)
  s = -1;                 F1
else {
  var i;
  s = 0;                  F2 } Fs
  i = 1;                  F3 }
  while (i <= n) {
    s = s+i;              F4 } Fb   Fw   Fv   Fe   Fc
    i = i+1;              F5 }
  }^F,T
}
```

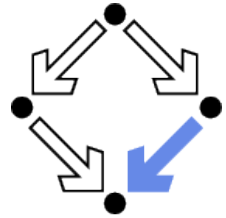
$F :\Leftrightarrow 1 <= \text{var } i <= \text{var } n+1 \text{ and var } s = \sum_{j=1}^{\text{var } i \text{-} 1} j$

$T := \text{var } n \text{ - var } i + 1$

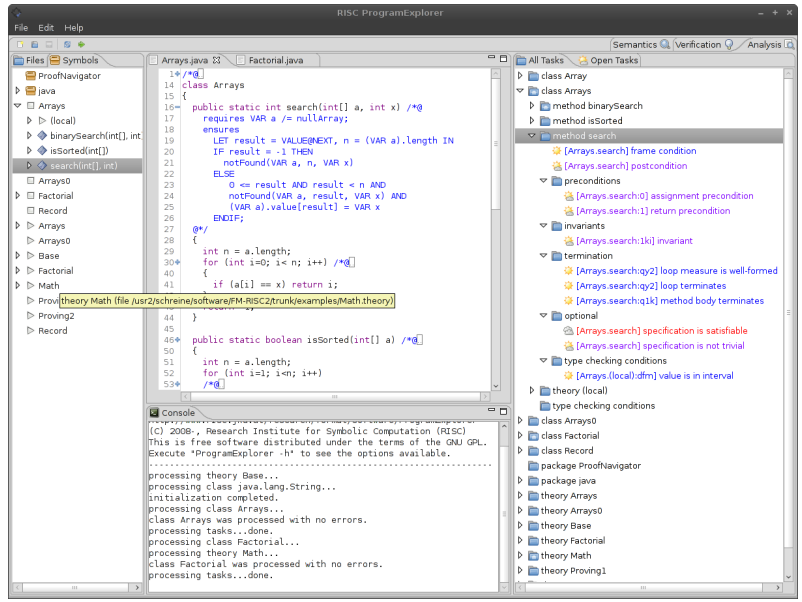
$$F_c \quad \Leftrightarrow \quad [\text{if old } n < 0 \text{ then var } s = \text{-}1 \text{ else var } s = \sum_{j=1}^{\text{old } n} j]^{\{s\}}$$

The program computes for non-negative n the sum from 1 to n.

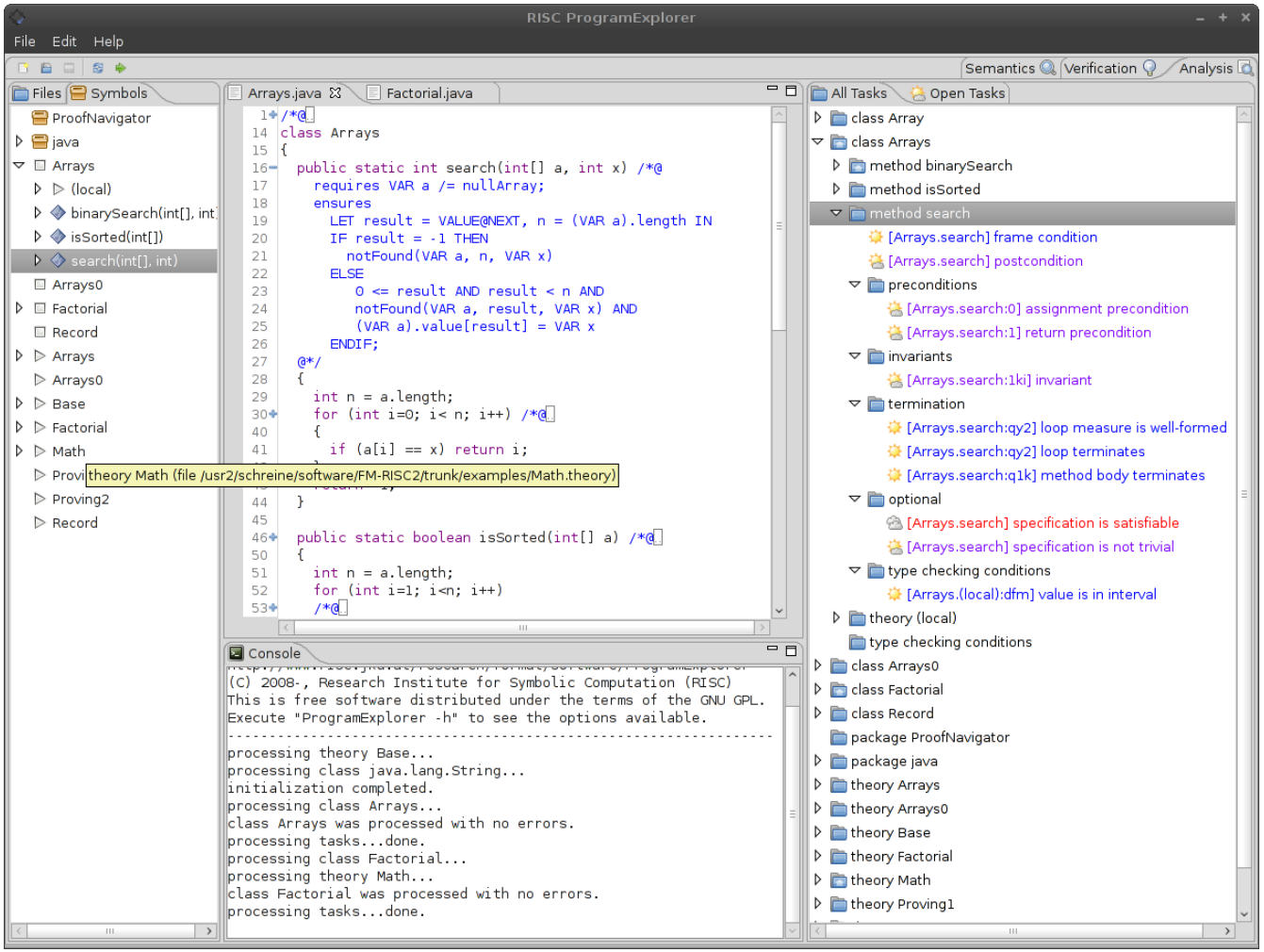
The RISC ProgramExplorer



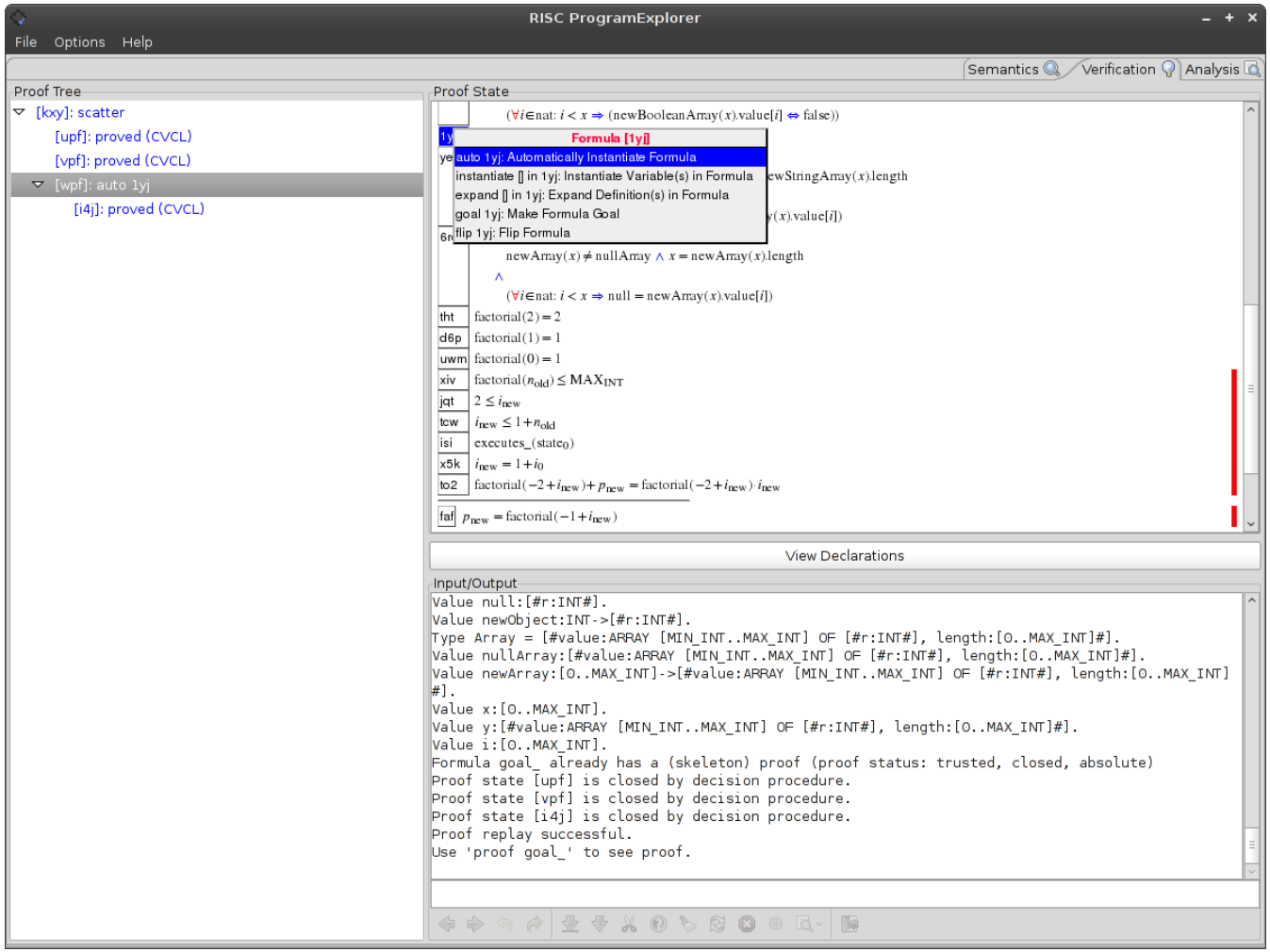
- An integrated program reasoning environment.
  - Programming language *MiniJava*.
  - Theory/specification language in the style of PVS/CVC.
  - Semi-automatic proving assistant *RISC ProofNavigator*.
- Semantics view.
  - Semantics of a method body.
  - Pre/post-condition reasoning.
- Analyze view (verification tasks).
  - Type checking conditions.
  - Statement preconditions.
  - Loop invariants.
  - Method frame preservation.
  - Method termination.
  - Method postcondition.
- Verify view.
  - Proof construction and management.



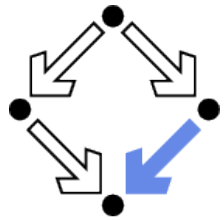
Demonstration



Demonstration



Current State and Further Work



- Development 2008–2011.
  - RISC ProofNavigator 2005–2007.
    - Uses CVCL as a validity checker.
  - About 140K lines of commented code.
- Software in beta status.
  - Functionality essentially complete.
  - Reasonably stable (tested with small examples only).
- Website and user manual.
  - Manual incomplete (status of April 2010).
- Current work:
  - Beta testing and documentation.
  - Use in "formal methods" course starting with fall 2011.

http://www.risc.jku.at/research/formal/software/ProgramExplorer