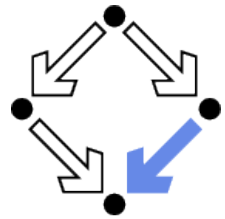
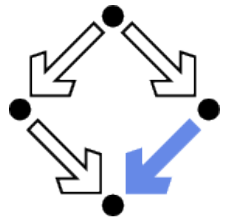
Computer-Assisted Program Reasoning Based on a Relational Semantics of Programs

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

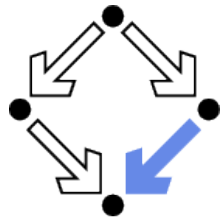
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.jku.at



Program Verification (Classical)



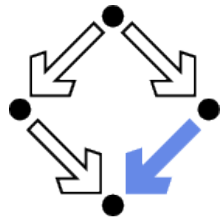
Program Verification (Classical)



P

- Program P

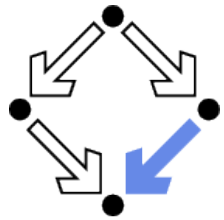
Program Verification (Classical)



$$P \quad + \quad S$$

- Program P
- Specification S

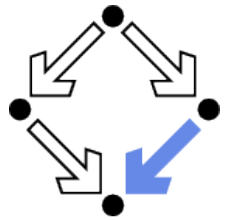
Program Verification (Classical)



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

- Program P
- Specification S
- Annotation A

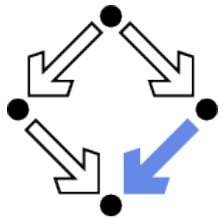
Program Verification (Classical)



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

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

Program Verification (Classical)

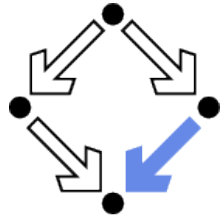


$$P^A \quad + \quad S \quad \begin{array}{c} \nearrow \\ \longrightarrow \\ \searrow \end{array} \quad \begin{array}{c} VC_1 \\ \vdots \\ VC_i \\ \vdots \\ VC_n \end{array}$$

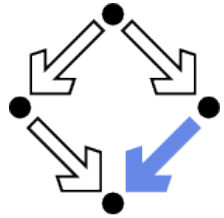
- Program P
- Specification S
- Annotation A
- Verification Conditions VC

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

Soundness of Verification

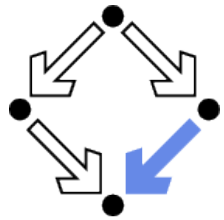


Soundness of Verification



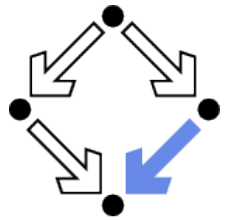
- Program P . . . command

Soundness of Verification



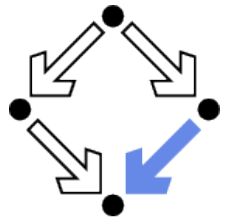
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$

Soundness of Verification



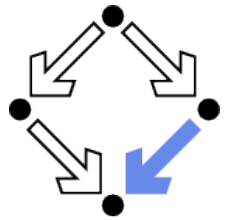
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula

Soundness of Verification



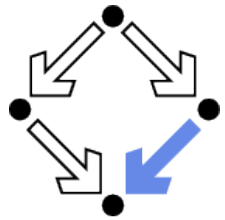
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$

Soundness of Verification



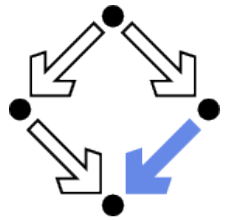
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$
- Verification Condition VC . . . formula

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 . . . formula
  - Condition semantics $[\![VC]\!] \in \{\text{true}, \text{false}\}$

Soundness of Verification

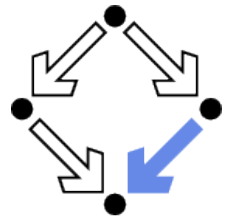


- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Specification S . . . formula
  - Specification semantics $[\![S]\!] \subseteq State \times State$
- Verification Condition VC . . . formula
  - Condition semantics $[\![VC]\!] \in \{\text{true}, \text{false}\}$

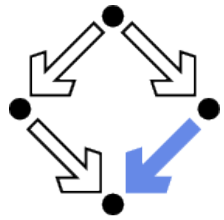
$$[\![VC]\!] \Rightarrow \forall s, s' \in State : [\![P]\!](s, s') \Rightarrow [\![S]\!](s, s')$$

Both the program P and the specification S denote state relations.

Program Reasoning (Alternative)



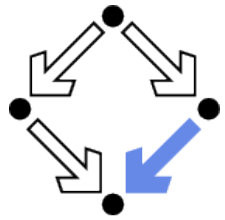
Program Reasoning (Alternative)



P^A

- Annotated program P^A

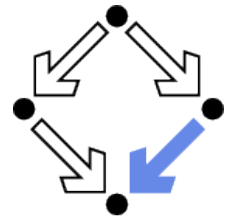
Program Reasoning (Alternative)



$$P^A \longrightarrow F$$

- Annotated program P^A
- Transition formula F

Program Reasoning (Alternative)

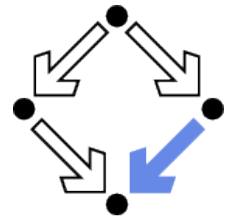


$$P^A \longrightarrow \circlearrowleft F$$

- Annotated program P^A
- Transition formula F

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

Program Reasoning (Alternative)

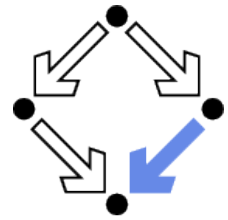


$$P^A \longrightarrow \circlearrowleft F \quad + \quad S$$

- 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)

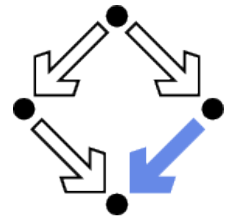


$$P^A \longrightarrow \overset{\circlearrowleft}{F} \quad + \quad S \longrightarrow F \Rightarrow S$$

- 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)



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

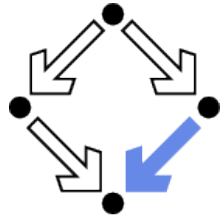
$$\searrow$$

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

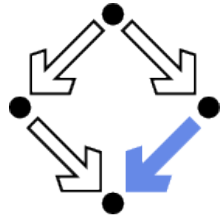
- Annotated program P^A
- Transition formula F
- Specification S
- Verification Condition $F \Rightarrow S$
- Translation Conditions TC

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

Soundness of Translation

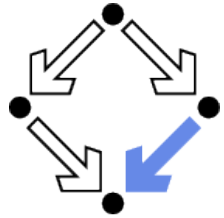


Soundness of Translation



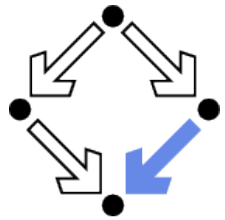
- Program P . . . command

Soundness of Translation



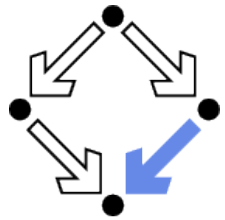
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$

Soundness of Translation



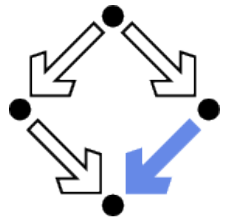
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula

Soundness of Translation



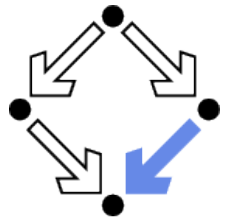
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$

Soundness of Translation



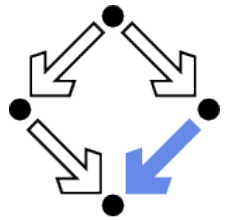
- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$
- Translation Conditions TC . . . formula

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 . . . formula
  - Condition semantics $[\![TC]\!] \in \{\text{true}, \text{false}\}$

Soundness of Translation

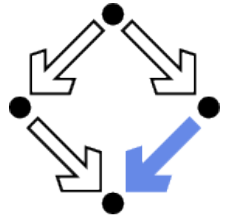


- Program P . . . command
  - Program semantics $[\![P]\!] \subseteq State \times State$
- Formula F . . . formula
  - Formula semantics $[\![F]\!] \subseteq State \times State$
- Translation Conditions TC . . . formula
  - Condition semantics $[\![TC]\!] \in \{\text{true}, \text{false}\}$

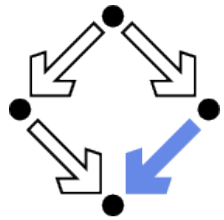
$$[\![TC]\!] \Rightarrow \forall s, s' \in State : [\![P]\!](s, s') \Rightarrow [\![F]\!](s, s')$$

Formula F captures the state relation denoted by program P.

Example



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
}
```

Example

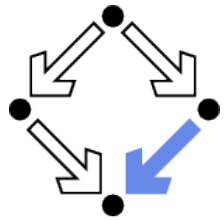
```
if (n < 0)
  s = -1;                 F1
else {
  var i;
   s = 0;                 F2 } Fs
   i = 1;                 F3 }
   while (i <= n) {
     s = s+i;             F4 } Fb
     i = i+1;             F5 }
   }^{F,T}
}
```

(Brackets: F_2, F_3 form F_s; F_4, F_5 form F_b; the loop is F_w; F_s and F_w form F_v; the else-block is F_e; the whole conditional is F_c.)

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

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

Example



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

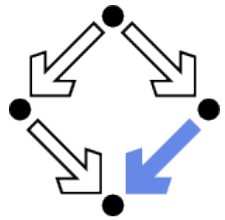
F_s = F_2, F_3; F_b = F_4, F_5; F_w = F_b, $F^{F,T}$; F_v = F_s, F_w; F_e = F_1, F_v; F_c = F_e

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

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

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

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
}
```

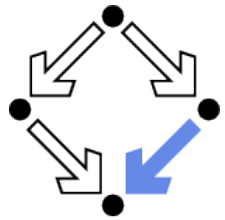
$F :\Leftrightarrow 1 <= \text{var } i <= \text{var } n{+}1 \text{ and var } s = \sum_{j=1}^{\text{var } i \text{ -1}} j$

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

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

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

Judgements



- Core judgement $c : [f_r]^{xs}_{g,h}$

  $\forall s, s' \in \mathit{State} :$
  $[\![g]\!] \wedge [\![h]\!](s) \Rightarrow$
  $([\![c]\!](s, s') \Rightarrow [\![f_r]\!](s, s') \wedge \forall x \in \mathit{Variable} \backslash xs : [\![x]\!](s) = [\![x]\!](s'))$.

  Command c, state relation f_r, set of program variables x_s,
  state-independent condition g, state condition h.

- Derived judgements $\mathrm{pre}(c, f_q) = f_p$ and $\mathrm{post}(c, f_p) = f_q$

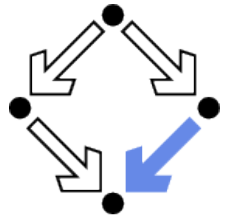
  $\forall s, s' \in \mathit{State} : [\![g]\!] \wedge [\![h]\!](s) \Rightarrow ([\![f_p]\!](s) \wedge [\![c]\!](s, s') \Rightarrow [\![f_q]\!](s'))$

$$\frac{c : [f]^{xs}_{g,h}}{\mathrm{pre}(c, f_q) = \forall xs : f[xs/\mathrm{var}\ xs] \Rightarrow f_q[xs/\mathrm{old}\ xs]}$$

$$\frac{c : [f]^{xs}_{g,h}}{\mathrm{post}(c, f_p) = \exists xs : f_p[xs/\mathrm{old}\ xs] \wedge f[xs/\mathrm{old}\ xs, \mathrm{old}\ xs/\mathrm{var}\ xs]}$$

A generalization of pre/post-condition reasoning.

Translation Calculus



$$\frac{e \simeq_h t}{x = e : [\text{var } x = t]^{\{x\}}_{\text{true},h}}$$

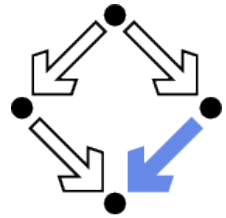
$$\frac{c : [f]^{xs}_{g,h}}{\{\text{var } x;\ c\} : [\exists x_0, x_1 : f[x_0/\text{old } x, x_1/\text{var } x]]^{xs \setminus x}_{g, \forall x : h[x/\text{old } x]}}$$

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

$$\frac{e \simeq_h f_e \qquad c_1 : [f_1]^{xs}_{g_1,h_1} \qquad c_2 : [f_2]^{xs}_{g_2,h_2}}{\text{if } (e) \text{ then } c_1 \text{ else } c_2 : [\text{if } f_e \text{ then } f_1 \text{ else } f_2]^{xs}_{g_1 \wedge g_2, h \wedge \text{if } f_e \text{ then } h_1 \text{ else } h_2}}$$

$$\frac{e \simeq_h f_e \qquad c : [f_c]^{xs}_{g_c,h_c} \qquad g \equiv \forall xs, ys, zs : f[xs/\text{old } xs, ys/\text{var } xs] \wedge f_e[ys/\text{old } xs] \wedge f_c[ys/\text{old } xs, zs/\text{var } xs] \Rightarrow h[ys/\text{old } xs] \wedge f[xs/\text{old } xs, zs/\text{var } xs]}{\text{while } (e)^{f,t}\ c : [f \wedge \neg f_e[\text{var } xs/\text{old } xs]]^{xs}_{g_c \wedge g, h \wedge f[\text{old } xs/\text{var } xs]}}$$

Termination



- Termination judgement $c \downarrow_{g_c} f_c$

$$\forall s \in \mathit{State} : [\![g_c]\!] \Rightarrow [\![f_c]\!](s) \Rightarrow \langle\!\langle c \rangle\!\rangle(s)$$

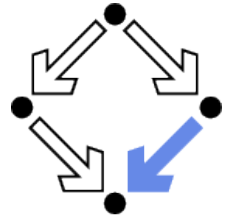
$$x = e \downarrow_{\mathsf{true}} \mathsf{true} \qquad \frac{c \downarrow_g f}{\{\mathtt{var}\ x;\ c\} \downarrow_g \forall x : f}$$

$$\frac{c_1 \downarrow_{g_1} f_1 \quad c_2 \downarrow_{g_2} f_2 \quad \mathsf{pre}(c_1, f_2) = f_3}{\{c_1 ; c_2\} \downarrow_{g_1 \wedge g_2} f_1 \wedge f_3}$$

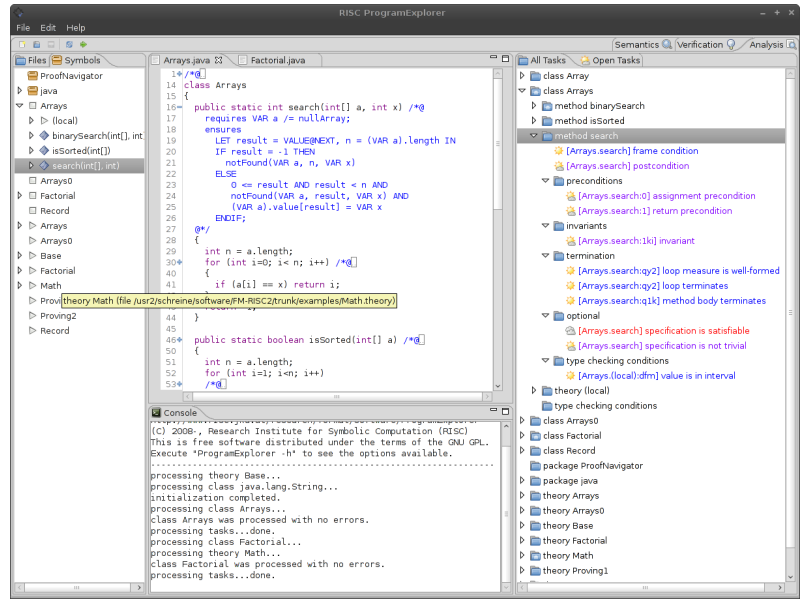
$$\frac{e \simeq_h f_e \qquad c_1 \downarrow_{g_1} f_1 \qquad c_2 \downarrow_{g_2} f_2}{\mathtt{if}\ (e)\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2 \downarrow_{g_1 \wedge g_2} \mathsf{if}\ f_e\ \mathsf{then}\ f_1\ \mathsf{else}\ f_2}$$

$$\frac{\begin{array}{c} e \simeq_h f_e \qquad c : [f_c]^{xs}_{g_c, h_c} \qquad c \downarrow_{g_t} f_t \\ g \equiv \forall xs, ys, zs : f[xs/\mathsf{old}\ xs, ys/\mathsf{var}\ xs] \wedge f_e[ys/\mathsf{old}\ xs] \wedge f_c[ys/\mathsf{old}\ xs, zs/\mathsf{var}\ xs] \Rightarrow \\ f_t[ys/\mathsf{old}\ xs] \wedge 0 \leqslant t[zs/\mathsf{old}\ xs] < t[ys/\mathsf{old}\ xs] \end{array}}{\mathtt{while}\ (e)^{f,t}\ c \downarrow_{g \wedge g_t} t \geqslant 0}$$

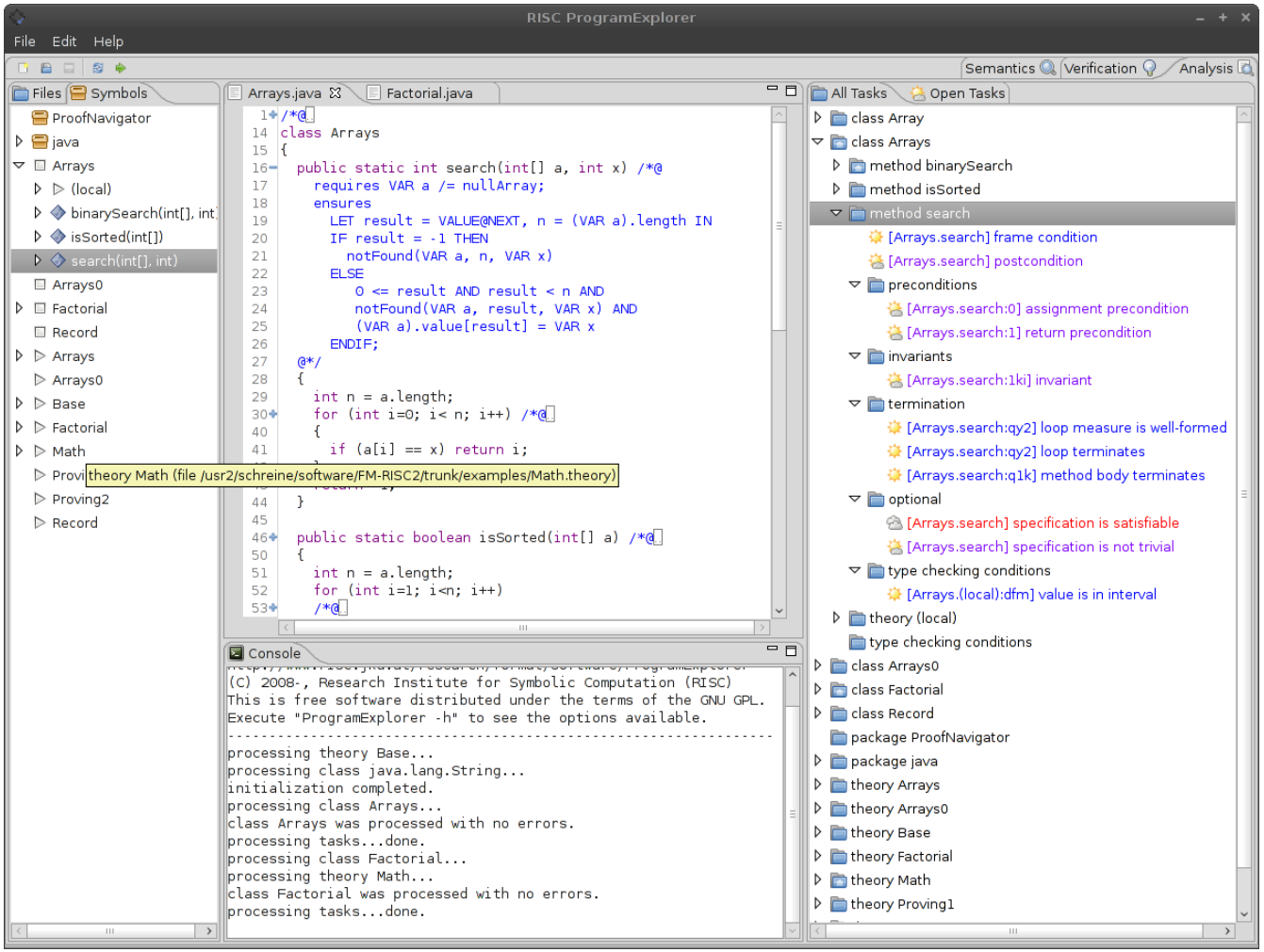
The RISC ProgramExplorer



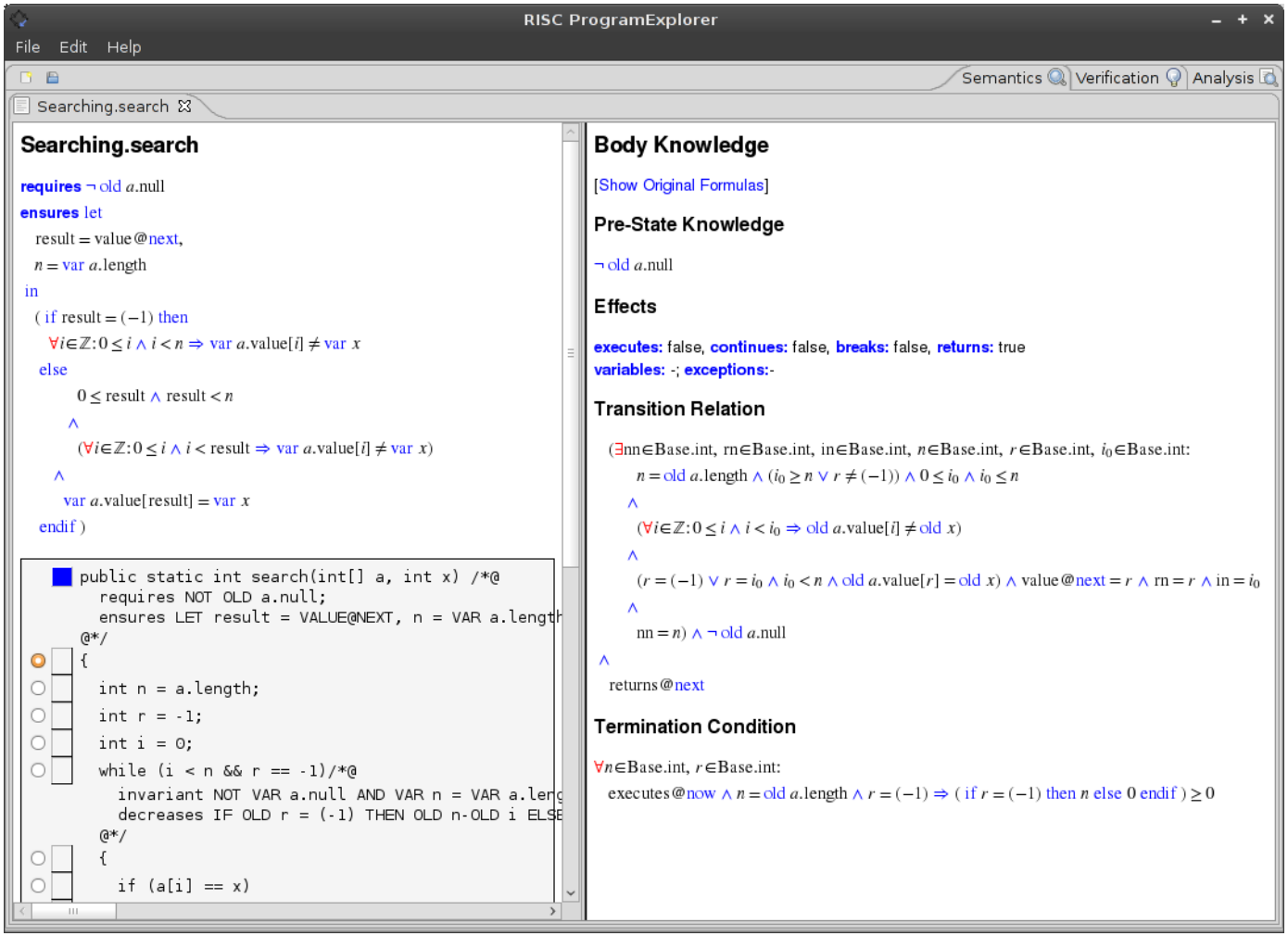
- An integrated program reasoning environment.
  - Programming language *MiniJava*.
  - Theory/specification language in the style of PVS/CVC.
  - Semi-automatic proving assistant *RISC ProofNavigator*.
- Analysis view (verification tasks).
  - Type checking conditions.
  - Statement preconditions.
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



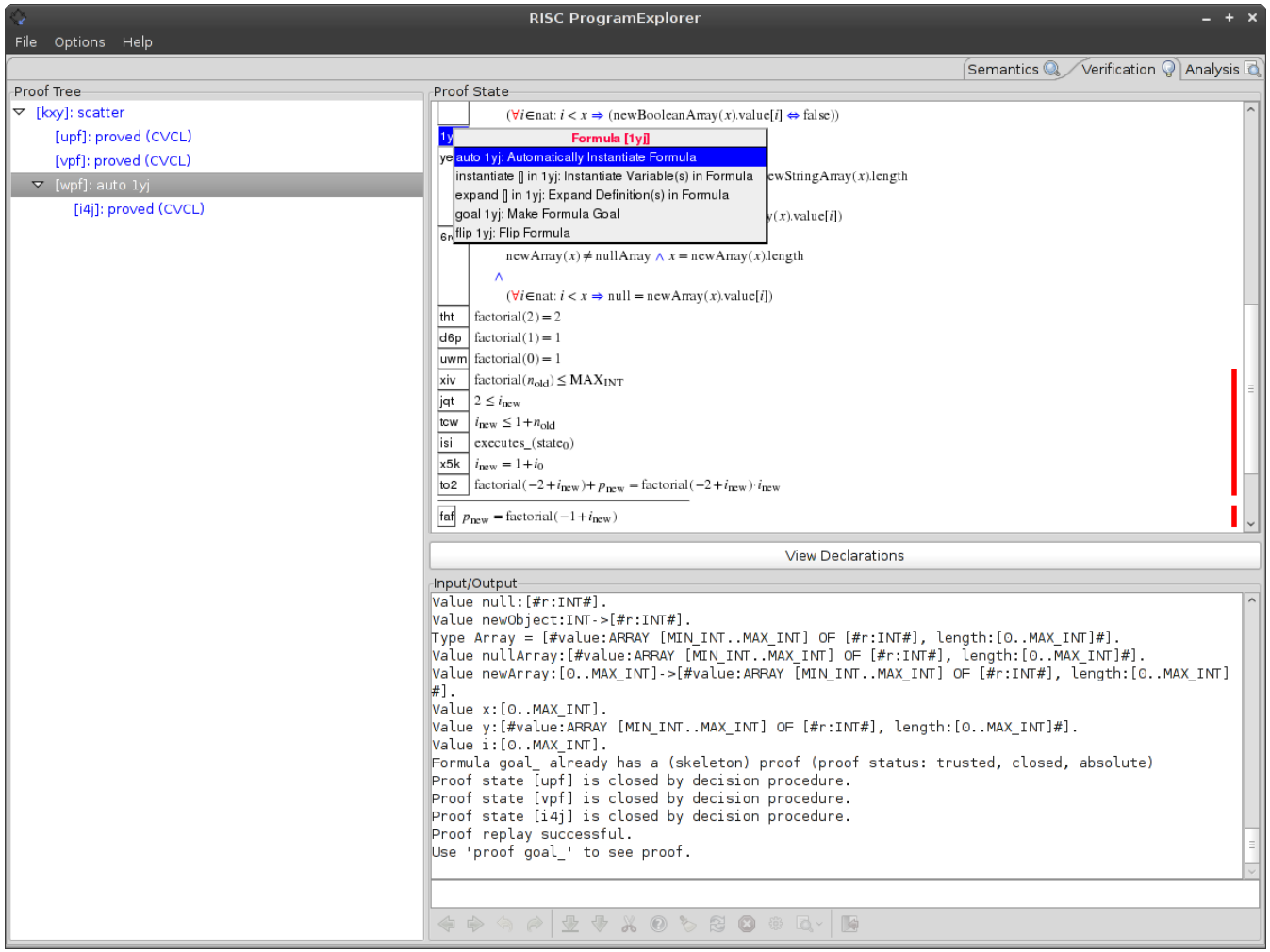
Analysis View



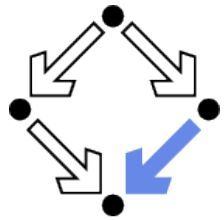
Semantics View



Verification View



Current State and Further Work



- Development 2008–2011.
  - RISC ProofNavigator 2005–2007.
    - Uses CVCL as a validity checker.
  - About 140K lines of commented Java code.
- October 2011: first release.
  - Tested with various examples.
  - Use in "formal methods" course starting with fall 2011.
- Future work:
  - Simplification of transition relations.
  - Improvement of proof automation.

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