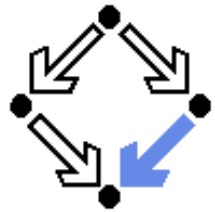
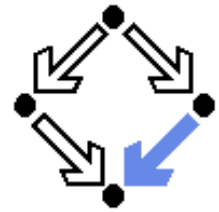
Reasoning about Control Flow Interruptions

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

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



Background



Past presentations: relational calculus for imperative programs.

- Command C:

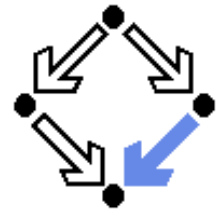
```
x = x/y;
```

- Formula F:

```
int(x) AND int(y) =>
  writesonly x AND int(x') AND
  (y /= 0 =>
    EXISTS $r: x = x'*y+$r AND
      0 <= $r AND $r < abs(y))
```

Valid judgement $C : F$.

Basic Idea



Extend calculus to treatment of interruptions.

- Command C:

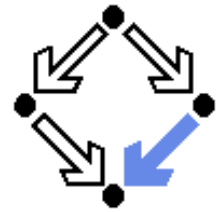
```
if (y == 0) throw DivisionByZero x;
x = x/y;
```

- Formula F:

```
int(x) AND int(y) =>
  writesonly x AND int(x') AND
  IF y = 0
    THEN next.throws DivisionByZero AND
         next.value = x AND x' = x
    ELSE next.executes AND
         EXISTS $r: x = x'*y+$r AND
           0 <= $r AND $r < abs(y)
```

Valid judgement $C : F$.

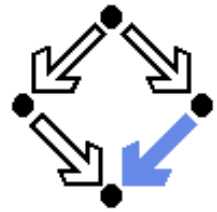
General Plan



Treatment of "non-standard" control flow in imperative programs.

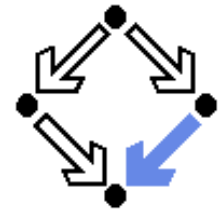
- Command language with control flow interruptions.
  - Language semantics: semantic domains, valuation functions.
- Formula language with control state predicates.
  - Language semantics: semantic domains, valuation functions.
- Verification calculus for judgements about commands.
  - Soundness of calculus.
- Adding loops to the language.
  - Syntax, semantics, judgements.
  - (Termination).
- (Pre- and post-condition reasoning).
- (Expressions and interruptions).

Extension of previously presented command language calculus.



1. **Command Language**
2. Formula Language
3. Verification Calculus
4. Loops
5. Conclusions

Command Language



- Abstract Syntax

$$C ::= \ldots$$
$$\mid \texttt{continue} \mid \texttt{break} \mid \texttt{return}\ E \mid \texttt{throw}\ I\ E$$
$$\mid \texttt{try}\ C_1\ \texttt{catch}(I_k\ I_v)\ C_2.$$

- Valuation Function

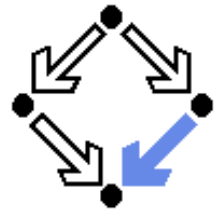
$$[\![\_]\!] : \mathbf{Command} \to \mathit{StateRelation}$$
$$[\![C]\!](s, s') \Leftrightarrow \ldots$$

- Semantic Domains

$$\mathit{StateRelation} := \mathbb{P}(\mathit{State} \times \mathit{State})$$
$$\mathit{State} := \ldots$$

States are not plain stores any more; now they also carry *control flow information.*

States



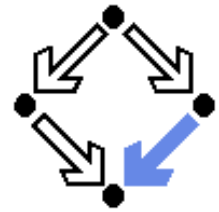
$$Flag := \{\mathrm{E}, \mathrm{C}, \mathrm{B}, \mathrm{R}, \mathrm{T}\}$$
$$Key := Value$$
$$Control := Flag \times Key \times Value$$
$$State := Store \times Control$$

$$store : State \to Store, store(s, c) = s$$
$$control : State \to Control, control(s, c) = c$$

$$flag : Control \to Flag, flag(f, k, v) = f$$
$$key : Control \to Key, key(f, k, v) = k$$
$$value : Control \to Value, value(f, k, v) = v$$

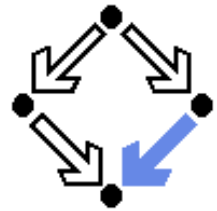
$$executes : \mathbb{P}(Control), executes(c) \Leftrightarrow flag(c) = \mathrm{E}$$
$$continues : \mathbb{P}(Control), continues(c) \Leftrightarrow flag(c) = \mathrm{C}$$
$$breaks : \mathbb{P}(Control), breaks(c) \Leftrightarrow flag(c) = \mathrm{B}$$
$$returns : \mathbb{P}(Control), returns(c) \Leftrightarrow flag(c) = \mathrm{R}$$
$$throws : \mathbb{P}(Control), throws(c) \Leftrightarrow flag(c) = \mathrm{T}$$

Changing State Flags

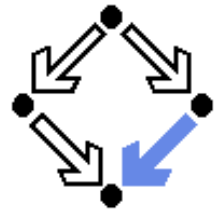


$$
\begin{aligned}
&\mathit{execute} : \mathit{State} \to \mathit{State} \\
&\mathit{execute}(s) = \\
&\quad \textsc{let}\ c = \mathit{control}(s)\ \textsc{in}\ (\mathit{store}(s), (\mathrm{E}, \mathit{key}(c), \mathit{value}(c)) \\
&\mathit{continue} : \mathit{State} \to \mathit{State} \\
&\mathit{continue}(s) = \\
&\quad \textsc{let}\ c = \mathit{control}(s)\ \textsc{in}\ (\mathit{store}(s), (\mathrm{C}, \mathit{key}(c), \mathit{value}(c)) \\
&\mathit{break} : \mathit{State} \to \mathit{State} \\
&\mathit{break}(s) = \\
&\quad \textsc{let}\ c = \mathit{control}(s)\ \textsc{in}\ (\mathit{store}(s), (\mathrm{B}, \mathit{key}(c), \mathit{value}(c)) \\
&\mathit{return} : \mathit{State} \times \mathit{Value} \to \mathit{State} \\
&\mathit{return}(s, v) = \\
&\quad \textsc{let}\ c = \mathit{control}(s)\ \textsc{in}\ (\mathit{store}(s), (\mathrm{R}, \mathit{key}(c), v)) \\
&\mathit{throw} : \mathit{State} \times \mathit{Key} \times \mathit{Value} \to \mathit{State} \\
&\mathit{throw}(s, k, v) = \\
&\quad \textsc{let}\ c = \mathit{control}(s)\ \textsc{in}\ (\mathit{store}(s), (\mathrm{B}, k, v))
\end{aligned}
$$

Valuation Functions

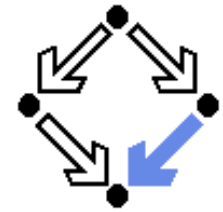


$$
\begin{aligned}
&[\![C_1 ; C_2]\!](s, s') \Leftrightarrow \\
&\quad \exists s_0 \in \mathit{State} : \\
&\qquad [\![C_1]\!](s, s_0) \wedge \\
&\qquad \text{IF } \mathit{executes}(\mathit{control}(s_0)) \text{ THEN } [\![C_2]\!](s_0, s') \text{ ELSE } s' = s_0 \\
&[\![\texttt{continue}]\!](s, s') \Leftrightarrow s' = \mathit{continue}(s) \\
&[\![\texttt{break}]\!](s, s') \Leftrightarrow s' = \mathit{break}(s) \\
&[\![\texttt{return } E]\!](s, s') \Leftrightarrow s' = \mathit{return}(s, [\![E]\!](s)) \\
&[\![\texttt{throw } I \ E]\!](s, s') \Leftrightarrow s' = \mathit{throw}(s, I, [\![E]\!](s)) \\
&[\![\texttt{try } C_1 \texttt{ catch(} I_k \ I_v \texttt{) } C_2]\!](s, s') \Leftrightarrow \\
&\quad \exists s_0, s_1, s_2 \in \mathit{State} : \\
&\qquad [\![C_1]\!](s, s_0) \wedge \\
&\qquad \text{IF } \mathit{throws}(\mathit{control}(s_0)) \wedge \mathit{key}(\mathit{control}(s_0)) = I_k \text{ THEN} \\
&\qquad\quad s_1 = \mathit{write}(\mathit{execute}(s_0), I_v, \mathit{value}(\mathit{control}(s_0))) \wedge \\
&\qquad\quad [\![C_2]\!](s_1, s_2) \wedge \\
&\qquad\quad s' = \mathit{write}(s_2, I_v, \mathit{read}(s_0, I_v)) \\
&\qquad \text{ELSE } s' = s_0
\end{aligned}
$$



1. Command Language
2. **Formula Language**
3. Verification Calculus
4. Loops
5. Conclusions

Formula Lanuguage with Control Predicates



$F \in \text{Formula}$

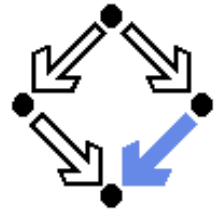
$S \in \text{State}$

$$
\begin{aligned}
F ::= \; & \ldots \\
& \mid \texttt{ALLSTATE}\ \#I_1, \ldots, \#I_n\texttt{:}\ F \\
& \mid \texttt{EXSTATE}\ \#I_1, \ldots, \#I_n\texttt{:}\ F \\
& \mid S_1\ \texttt{==}\ S_2 \\
& \mid S\texttt{.executes} \mid S\texttt{.continues} \mid S\texttt{.breaks} \\
& \mid S\texttt{.returns} \mid S\texttt{.throws} \mid S\texttt{.throws}\ I
\end{aligned}
$$

$$T ::= \ldots \mid S\texttt{.value}$$

$$S ::= \texttt{now} \mid \texttt{next} \mid \#I$$

Language Semantics



- Semantic domains and operations

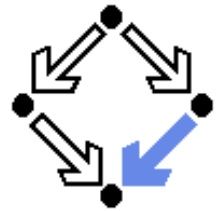
$$\mathit{ControlEnv} := \mathrm{Identifier} \to \mathit{Control}$$
$$\mathit{Environment} := \mathit{ValueEnv} \times \mathit{ControlEnv}$$

$$(e_v, e_c)(I) \equiv e_v(I)$$
$$(e_v, e_c)(I)_c \equiv e_c(I)$$
$$(e_v, e_c)[\, I_1 \mapsto v_1, \ldots, I_n \mapsto v_n \,] \equiv$$
$$\quad (e_v[\, I_1 \mapsto v_1, \ldots, I_n \mapsto v_n \,], e_c)$$
$$(e_v, e_c)[\, I_1 \mapsto c_1, \ldots, I_n \mapsto c_n \,]_c \equiv$$
$$\quad (e_v, e_c[\, I_1 \mapsto c_1, \ldots, I_n \mapsto c_n \,])$$

- Valuation functions

$$[\![\_]\!] : \mathrm{Formula} \to \mathit{Environment} \to \mathit{StateRelation}$$
$$[\![F]\!](e)(s, s') \Leftrightarrow \ldots$$
$$[\![\_]\!] : \mathrm{Term} \to \mathit{Environment} \to (\mathit{State} \times \mathit{State}) \to \mathit{Value}$$
$$[\![T]\!](e)(s, s') = \ldots$$
$$[\![\_]\!] : \mathrm{State} \to \mathit{Environment} \to (\mathit{State} \times \mathit{State}) \to \mathit{Control}$$
$$[\![S]\!](e)(s, s') = \ldots$$

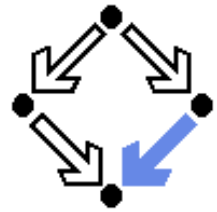
Valuation Functions



$$[\![\texttt{ALLSTATE}\ \#I_1, \ldots, \#I_n : F]\!](e)(s, s') \Leftrightarrow$$
$$\quad \forall c_1, \ldots, c_n \in \mathit{Control} : [\![F]\!](e[I_1 \mapsto c_1, \ldots, I_n \mapsto c_n]_c)(s, s')$$
$$[\![\texttt{EXSTATE}\ \#I_1, \ldots, \#I_n : F]\!](e)(s, s') \Leftrightarrow$$
$$\quad \exists c_1, \ldots, c_n \in \mathit{Control} : [\![F]\!](e[I_1 \mapsto c_1, \ldots, I_n \mapsto c_n]_c)(s, s')$$
$$[\![S_1 == S_2]\!](e)(s, s') \Leftrightarrow [\![S_1]\!](e)(s, s') = [\![S_2]\!](e)(s, s')$$
$$[\![S\texttt{.executes}]\!](e)(s, s') \Leftrightarrow \mathit{executes}([\![S]\!](e)(s, s'))$$
$$[\![S\texttt{.continues}]\!](e)(s, s') \Leftrightarrow \mathit{continues}([\![S]\!](e)(s, s'))$$
$$[\![S\texttt{.breaks}]\!](e)(s, s') \Leftrightarrow \mathit{breaks}([\![S]\!](e)(s, s'))$$
$$[\![S\texttt{.returns}]\!](e)(s, s') \Leftrightarrow \mathit{returns}([\![S]\!](e)(s, s'))$$
$$[\![S\texttt{.throws}]\!](e)(s, s') \Leftrightarrow \mathit{throws}([\![S]\!](e)(s, s'))$$
$$[\![S\texttt{.throws}\ I]\!](e)(s, s') \Leftrightarrow$$
$$\quad \textsc{let}\ c = [\![S]\!](e)(s, s')\ \textsc{in}\ \mathit{throws}(c) \wedge \mathit{key}(c) = I$$

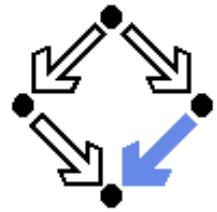
$$[\![S\texttt{.value}]\!](e)(s, s') = \mathit{value}([\![S]\!](e)(s, s'))$$

$$[\![\texttt{now}]\!](e)(s, s') = \mathit{control}(s)$$
$$[\![\texttt{next}]\!](e)(s, s') = \mathit{control}(s')$$
$$[\![\#I]\!](e)(s, s') = e(I)_c$$



1. Command Language
2. Formula Language
3. **Verification Calculus**
4. Loops
5. Conclusions

Verification Calculus



- Judgement:

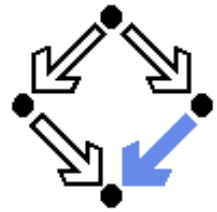
$$C : F \Leftrightarrow$$
$$\forall s, s' \in \mathit{State}, e \in \mathit{Environment} :$$
$$[\![\texttt{now.executes}]\!](e)(s, s') \wedge [\![C]\!](s, s') \Rightarrow$$
$$[\![F]\!](e)(s, s')$$

- Definition:

$$[F]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n} \equiv$$
$$(F) \texttt{ AND writesonly } I_1, \ldots, I_n \texttt{ AND}$$
$$\texttt{(next.continues => } (F_c)\texttt{) AND}$$
$$\texttt{(next.breaks => } (F_b)\texttt{) AND}$$
$$\texttt{(next.returns => } (F_r)\texttt{) AND}$$
$$\texttt{(next.throws =>}$$
$$\texttt{(next.throws } K_1 \texttt{ OR ... OR next.throws } K_n\texttt{))}$$

Judgements will have form $C : [F]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n}$.

Verification Calculus



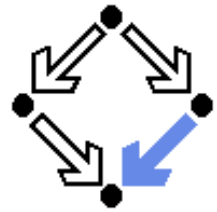
$$\frac{C : [\, F \,]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n} \quad \text{p is a permutation of } \{1, \ldots, n\}}{C : [\, F \,]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_{p(1)}, \ldots, I_{p(n)}}}$$

$$\frac{C : [\, F \,]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n} \quad I \neq I_1 \wedge \ldots \wedge I \neq I_n}{C : [\, F \texttt{ AND } I' = I \,]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n, I}}$$

$$\frac{C : [\, F \,]^{\texttt{FALSE}, \texttt{FALSE}, \texttt{FALSE}, \emptyset}_{I_1, \ldots, I_n}}{C : [\, F \,]_{I_1, \ldots, I_n}}$$

$$\frac{C \text{ is a program without interruptions} \quad C : [\, F \,]_{I_1, \ldots, I_n}}{C : [\, F \,]^{\texttt{FALSE}, \texttt{FALSE}, \texttt{FALSE}, \emptyset}_{I_1, \ldots, I_n}}$$

Verification Calculus

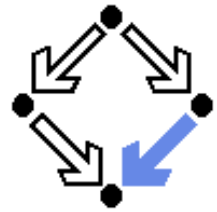


$$\frac{E \simeq T}{I = E : [\, I' = T \ \texttt{AND}\ \texttt{next.executes} \,]_{I}^{\texttt{FALSE},\texttt{FALSE},\texttt{FALSE},\emptyset}}$$

$$\frac{\begin{array}{l} C : [\, F \,]_{I_1,\ldots,I_n,I}^{F_c,F_b,F_r,\{K_1,\ldots,K_m\}} \\ I_a \neq I_b \end{array}}{\begin{array}{l} \texttt{var}\ I;\ C : \\ \quad [\,\texttt{EXISTS}\ \$I_a, \$I_b : F[\$I_a/I, \$I_b/I']\,]_{I_1,\ldots,I_n}^{F_c,F_b,F_r,\{K_1,\ldots,K_m\}} \end{array}}$$

$$\frac{\begin{array}{l} C : [\, F \,]_{I_1,\ldots,I_n,I}^{F_c,F_b,F_r,\{K_1,\ldots,K_m,\}} \\ I_a \neq I_b \\ E \simeq T \end{array}}{\begin{array}{l} \texttt{var}\ I{=}E;\ C : \\ \quad [\,\texttt{EXISTS}\ \$I_a, \$I_b : \\ \qquad \$I_a{=}T\ \texttt{AND}\ F[\$I_a/I, \$I_b/I']\,]_{I_1,\ldots,I_n}^{F_c,F_b,F_r,\{K_1,\ldots,K_m\}} \end{array}}$$

Verification Calculus



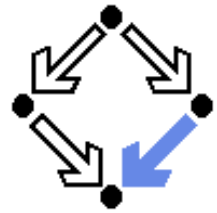
$$\texttt{continue} : [\texttt{next.continues}]_{\_}^{\texttt{TRUE},\texttt{FALSE},\texttt{FALSE},\emptyset}$$

$$\texttt{break} : [\texttt{next.breaks}]_{\_}^{\texttt{FALSE},\mathrm{t}t\mathit{TRUE},\texttt{FALSE},\emptyset}$$

$$\frac{T \simeq E}{\begin{array}{l}\texttt{return } E : \\ \quad [\texttt{next.returns} \\ \quad \texttt{AND next.value=}T\,]_{\_}^{\texttt{FALSE},\texttt{FALSE},\texttt{TRUE},\emptyset}\end{array}}$$

$$\frac{T \simeq E}{\begin{array}{l}\texttt{throw } I\ E : \\ \quad [\texttt{next.throws } I \\ \quad \texttt{AND next.value=}T\,]_{\_}^{\texttt{FALSE},\texttt{FALSE},\texttt{FALSE},\{I\}}\end{array}}$$

Verification Calculus



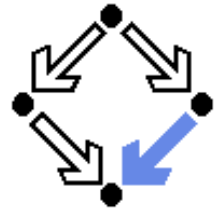
$$\frac{C_1 : [\,F_1\,]^{\texttt{FALSE},\texttt{FALSE},\texttt{FALSE},\emptyset}_{I_1,\ldots,I_n} \qquad C_2 : [\,F_2\,]^{F_c,F_b,F_r,\{K_1,\ldots,K_m\}}_{I_1,\ldots,I_n}}{C_1\,;C_2 :}$$

$$[\,\texttt{EXISTS } \$I_1,\ldots,\$I_n\texttt{: EXSTATE } \#I_s\texttt{:}$$
$$F_1[\#I_s/\texttt{next}][\$I_1/I_1\texttt{'},\ldots,\$I_n/I_n\texttt{'}]\ \texttt{AND}$$
$$F_2[\#I_s/\texttt{now}][\$I_1/I_1,\ldots,\$I_n/I_n]\,]^{F_c,F_b,F_r,\{K_1,\ldots,K_m\}}_{I_1,\ldots,I_n}$$

$$\frac{C_1 : [\,F_1\,]^{F_{c1},F_{b1},F_{r1},\{K_1,\ldots,K_m\}}_{I_1,\ldots,I_n} \qquad C_2 : [\,F_2\,]^{F_{c1},F_{b1},F_{r1},\{L_1,\ldots,L_o\}}_{I_1,\ldots,I_n}}{C_1\,;C_2 :}$$

$$[\,\texttt{EXISTS } \$I_1,\ldots,\$I_n\texttt{: EXSTATE } \#I_s\texttt{:}$$
$$F_1[\#I_s/\texttt{next}][\$I_1/I_1\texttt{'},\ldots,\$I_n/I_n\texttt{'}]\ \texttt{AND}$$
$$\texttt{IF } \#I_s\texttt{.executes THEN}$$
$$F_2[\#I_s/\texttt{now}][\$I_1/I_1,\ldots,\$I_n/I_n]$$
$$\texttt{ELSE}$$
$$I_1\texttt{'=}\$I_1\ \texttt{AND}\ \ldots\ \texttt{AND}\ I_n\texttt{'=}\$I_n\ \texttt{AND next==}\#I_s$$
$$]^{F_{c1}\ \texttt{OR}\ F_{c2},F_{b1}\ \texttt{OR}\ F_{b2},F_{r1}\ \texttt{OR}\ F_{r2},\{K_1,\ldots,K_m,L_1,\ldots,L_o\}}_{I_1,\ldots,I_n}$$

Example (Before Simplification)

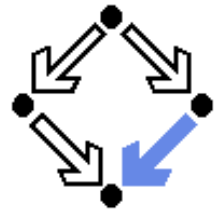


$C : [\, F \,]^{\texttt{FALSE},\texttt{FALSE},\texttt{FALSE},\{\texttt{DivByZero}\}}_{\texttt{y},\texttt{z}}$

```
if (x=0) throw DivByZero y else y = y/x;
z=z+y;
```

```
EXISTS $y, $z: EXSTATE #s:
    IF x=0 THEN
      $y=y AND $z=z AND
      #s.throws DivByZero AND #s.value = y
    ELSE
      $y=y/x AND $z=z AND #s.executes
  AND
    IF #s.executes THEN
      y'=$y AND z'=$z+$y AND next.executes
    ELSE
      y'=$y AND z'=$z AND next==#s
```

Example (After Simplification)



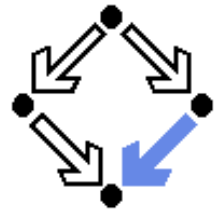
$C : [\, F \,]^{\texttt{FALSE},\texttt{FALSE},\texttt{FALSE},\{\texttt{DivByZero}\}}_{\texttt{y},\texttt{z}}$

```
if (x=0) throw DivByZero y else y = y/x;
z=z+y;
```

```
IF x=0 THEN
  y'=y AND z'=z AND
  next.throws DivByZero AND next.value=y
ELSE
  y'=y/x AND z'=z+y/x AND
  next.executes
```

Specification expresses *essence* of program.

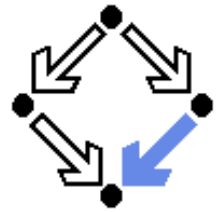
Verification Calculus



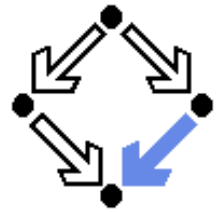
$$\frac{C : [\, F \,]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n} \qquad E \simeq F_0}{\texttt{if (}E\texttt{) } C : [\, \texttt{IF } F_0 \texttt{ THEN } F \texttt{ ELSE readsonly} \,]^{F_c, F_b, F_r, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n}}$$

$$\frac{C_1 : [\, F_1 \,]^{F_{c1}, F_{b1}, F_{r1}, \{K_1, \ldots, K_m\}}_{I_1, \ldots, I_n} \qquad C_2 : [\, F_2 \,]^{F_{c2}, F_{b2}, F_{r2}, \{L_1, \ldots, L_o\}}_{I_1, \ldots, I_n} \qquad E \simeq F_0}{\begin{array}{l}\texttt{if (}E\texttt{) } C_1 \texttt{ else } C_2 : \\ \quad [\, \texttt{IF } F_0 \texttt{ THEN } F_1 \\ \qquad \texttt{ELSE } F_2 \,]^{F_{c1} \texttt{ OR } F_{c2}, F_{b1} \texttt{ OR } F_{b2}, F_{r1} \texttt{ OR } F_{r2}, \{K_1, \ldots, K_m, L_1, \ldots, L_o\}}_{I_1, \ldots, I_n}\end{array}}$$

Verification Calculus

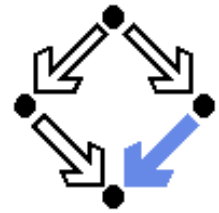


$$\frac{\begin{array}{l} C_1 : [\, F_1 \,]^{F_{c1},F_{b1},F_{r1},\{K_1,\ldots,K_m\}}_{I_1,\ldots,I_n} \\ C_2 : [\, F_2 \,]^{F_{c2},F_{b2},F_{r2},\{L_1,\ldots,L_o\}}_{I_1,\ldots,I_n} \\ I_a \neq I_b \\ \{I_a, I_b\} \cap \{I_1, \ldots, I_n\} = \emptyset \end{array}}{\begin{array}{l} \texttt{try } C_1 \texttt{ catch}(I_k\ I_v)\ C_2 : \\ \quad [\texttt{EXISTS \$}I_1,\ldots,\texttt{\$}I_n\texttt{: EXSTATE \#}I_s\texttt{:} \\ \quad\quad F_1[\texttt{\#}I_s/\texttt{next}][\texttt{\$}I_1/I_1\texttt{'},\ldots,\texttt{\$}I_n/I_n\texttt{'}]\ \texttt{AND} \\ \quad\quad \texttt{IF \#}I_s\texttt{.throws }I_k\texttt{ THEN} \\ \quad\quad\quad \texttt{EXISTS \$}I_a,\texttt{\$}I_b\texttt{: EXSTATE \#}I_t\texttt{:} \\ \quad\quad\quad\quad \texttt{\$}I_a\texttt{ = \#}I_s\texttt{.value AND \#}I_t\texttt{.executes AND} \\ \quad\quad\quad\quad F_2[\texttt{\#}I_t/\texttt{now}][\texttt{\$}I_a/I_v][\texttt{\$}I_1/I_1,\ldots,\texttt{\$}I_n/I_n][\texttt{\$}I_b/I_v\texttt{'}] \\ \quad\quad \texttt{ELSE} \\ \quad\quad\quad I_1\texttt{'=\$}I_1\texttt{ AND ... AND }I_n\texttt{'=\$}I_n\texttt{ AND next==\#}I_s \\ \quad]^{F_{c1}\ \texttt{OR}\ F_{c2},F_{b1}\ \texttt{OR}\ F_{b2},F_{r1}\ \texttt{OR}\ F_{r2},(\{K_1,\ldots,K_m\}\setminus\{I_k\})\cup\{L_1,\ldots,L_o\}}_{I_1,\ldots,I_n} \end{array}}$$



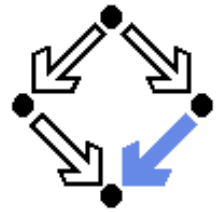
1. Command Language
2. Formula Language
3. Verification Calculus
4. **Loops**
5. Conclusions

Loops



$$
\begin{array}{l}
\mathit{finiteExecution}: \\
\mathbb{P}(\mathbb{N} \times \mathit{State}^\infty \times \mathit{State}^\infty \times \mathit{State} \times \mathit{StateFunction} \times \mathit{StateRelation}) \\
\mathit{finiteExecution}(k, t, u, s, E, C) \Leftrightarrow \\
\quad t(0) = s \wedge u(0) = s \wedge \\
\quad \forall i \in \mathbb{N}_k : \\
\qquad \neg \mathit{breaks}(\mathit{control}(u(i))) \wedge \mathit{executes}(\mathit{control}(t(i))) \wedge \\
\qquad E(t(i)) = \textsc{true} \wedge C(t(i), u(i+1)) \wedge \\
\qquad \textsc{if}\ \mathit{continues}(\mathit{control}(u(i+1))) \vee \mathit{breaks}(\mathit{control}(u(i+1))) \\
\qquad\quad \textsc{then}\ t(i+1) = \mathit{execute}(u(i+1)) \\
\qquad\quad \textsc{else}\ t(i+1) = u(i+1) \\
[\![\texttt{while (}E\texttt{) } C]\!](s, s') \Leftrightarrow \\
\quad \exists k \in \mathbb{N}, t, u \in \mathit{State}^\infty : \\
\qquad \mathit{finiteExecution}(k, t, u, s, [\![E]\!], [\![C]\!]) \wedge \\
\qquad ([\![E]\!](t(k)) \neq \textsc{true} \vee \\
\qquad\quad \neg(\mathit{executes}(\mathit{control}(u(k))) \vee \\
\qquad\qquad \mathit{continues}(\mathit{control}(u(k))))) \wedge \\
\qquad t(k) = s'
\end{array}
$$

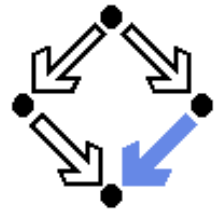
Verification Calculus



$$\frac{\begin{array}{l} C : [\,F\,]^{F_c,F_b,F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n} \\ E \simeq H \end{array}}{\begin{array}{l} \texttt{while (}E\texttt{) }C\texttt{ :} \\ \quad [\,\texttt{!next.continues AND} \\ \qquad \texttt{!next.breaks}\,]^{\texttt{FALSE},\texttt{FALSE},F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n} \end{array}}$$

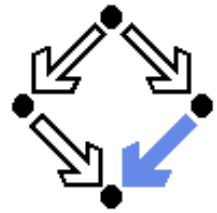
$$\frac{\begin{array}{l} C : [\,F\,]^{F_c,\texttt{FALSE},F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n} \\ E \simeq H \end{array}}{\begin{array}{l} \texttt{while (}E\texttt{) }C\texttt{ :} \\ \quad [\,\texttt{!next.continues AND !next.breaks AND} \\ \quad \texttt{(next.executes =>} \\ \qquad \texttt{!}H[\texttt{next}/\texttt{now}][I_1\texttt{'}/I_1,\ldots,I_n\texttt{'}/I_n]\texttt{)} \\ \quad]^{\texttt{FALSE},\texttt{FALSE},F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n} \end{array}}$$

Verification Calculus



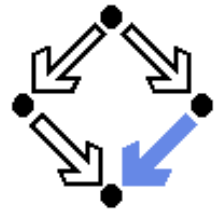
$$
\begin{aligned}
&\mathit{Invariant}(G, H, F)_{I_1,\ldots,I_n} \equiv \\
&\quad G \text{ has no free (mathematical or state) variables} \wedge \\
&\quad \forall e \in \mathit{ValueEnv}, s, s' \in \mathit{Store} : \\
&\qquad [\![\texttt{FORALL } \$I_1, \ldots, \$I_n \texttt{: ALLSTATE } \#I_s, \#I_t : \\
&\qquad\quad (G[\#I_s/\texttt{next}][\$I_1/I_1\texttt{'}, \ldots, \$I_n/I_n\texttt{'}] \\
&\qquad\qquad \texttt{AND } \#I_s\texttt{.executes} \\
&\qquad\qquad \texttt{AND } H[\#I_s/\texttt{now}][\$I_1/I_1, \ldots, \$I_n/\ \mathit{prevar}\, I_n] \\
&\qquad\qquad \texttt{AND } F[\#I_s/\texttt{now}, \#I_t/\texttt{next}][\$I_1/I_1, \ldots, \$I_n/I_n] \\
&\qquad\qquad \texttt{AND IF } \#I_t\texttt{.breaks OR } \#I_t\texttt{.continues} \\
&\qquad\qquad\quad \texttt{THEN next.executes} \\
&\qquad\qquad\quad \texttt{ELSE } \#I_t \texttt{ == next}) \texttt{ => } G) \,]\!](e)(s, s')
\end{aligned}
$$

Verification Calculus



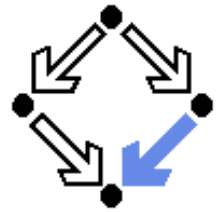
$$\frac{C : [\,F\,]^{F_c,F_b,F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n} \quad E \simeq H \quad \mathit{Invariant}(G,H,F)_{I_1,\ldots,I_n}}{\begin{array}{l}\texttt{while } (E)\ C : \\ \quad [\,\texttt{!next.continues AND !next.breaks AND} \\ \quad (G[\texttt{now/next}][I_1/I_1\texttt{'},\ldots,I_n/I_n\texttt{'}] \texttt{ =>} \\ \qquad G)\,]^{\texttt{FALSE},\texttt{FALSE},F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n}\end{array}}$$

$$\frac{C : [\,F\,]^{F_c,\texttt{FALSE},F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n} \quad E \simeq H \quad \mathit{Invariant}(G,H,F)_{I_1,\ldots,I_n}}{\begin{array}{l}\texttt{while } (E)\ C : \\ \quad [\,\texttt{!next.continues AND !next.breaks AND} \\ \quad (\texttt{next.executes =>} \\ \qquad \texttt{!}H[\texttt{next/now}][I_1\texttt{'}/I_1,\ldots,I_n\texttt{'}/I_n])\ \texttt{AND} \\ \quad (G[\texttt{now/next}][I_1/I_1\texttt{'},\ldots,I_n/I_n\texttt{'}] \texttt{ =>} \\ \qquad G)\,]^{\texttt{FALSE},\texttt{FALSE},F_r;K_1,\ldots,K_m}_{I_1,\ldots,I_n}\end{array}}$$



1. Command Language
2. Formula Language
3. Verification Calculus
4. Loops
5. **Conclusions**

Conclusions



- Soundness proofs of presented rules completed.
- Current work: extension of work to termination calculus.
- Next: pre- and post-condition reasoning.
- Next: expressions and interruptions.
- Finally: methods.

Hope to complete calculus by end of semester.