Choose your weapon - entry level

Bernhard Schätz - Perlen der Informatik
Example: Synchronized Producer Consumer

- **wait**: produces, exchanges
- **write**: reads, produces, exchanges
- **read**: exchanges, consumes
- **ready**: exchanges, consumes
- **consume**: waits, ready
- **exchange**: waits, ready, exchanges
- **produce**: waits, reads, exchanges
- **write, read**: produces, exchanges
Models

Classes of models:

- **Algebraic model**: Specification in form of syntactic formulae
  - Concepts: Process
  - Verification: Term equivalence
  - Example: ACP
- **Operational model**: Specification in form of abstract machine
  - Concepts: State, (labeled) transition
  - Verification: (Bi-)Simulation
  - Example: CCS
- **Denotational model**: Specification in form of observable behavior
  - Concepts: Event, (observation) trace
  - Verification: Behavior Inclusion
  - Example: TCSP
Algebraic Semantics: Syntax

Syntax: Process terms over alphabet $A$

- Blocking process: $0$
- Action prefix for some $a$ of $A$: $a.P$
- Alternative choice: $P + Q$
- Parallel composition: $P | Q$
- (Mutual) Recursion: $P = Q$

\[
S = p.x.S
\]
\[
R = \overline{x}.c.R
\]
\[
E = p.\tau.B
\]
\[
B = c.E + p.F
\]
\[
P = S | R
\]
\[
F = c.\tau.B
\]
Algebraic Semantics: Rules

\[ \begin{align*}
0 + P &= P \\
P + P &= P \\
P + Q &= Q + P \\
P + (Q + R) &= (P + Q) + R \\
0 | P &= 0 \\
P | Q &= Q | P \\
P | (Q | R) &= (P | Q) | R \\
a.P | \overline{a}.Q &= \tau.P | Q \\
a.P | b.Q &= a.(P | b.Q) + b.(a.P | Q) \\
\end{align*} \]

\[ P = S[P] \land Q = T[Q] \land P = Q \Rightarrow S[P] = T[Q] \]

\[ P = Q \]

Algebraic semantics:
- Relates equivalent processes
- Defined as equations of process terms
Algebraic Semantics: Example

Example: Proving equivalence of process terms $S_1$ and $S_2$

- Approach:
  - Construct chain of equations
  - Use inductive principles for recursion
Classes of models:

- **Algebraic model**: Specification in form of syntactic formulae  
  - Concepts: Process  
  - Verification: Term equivalence  
  - Example: ACP

- **Operational model**: Specification in form of abstract machine  
  - Concepts: State, (labeled) transition  
  - Verification: (Bi-)Simulation  
  - Example: CCS

- **Denotational model**: Specification in form of observable behavior  
  - Concepts: Event, (observation) trace  
  - Verification: Behavior Inclusion  
  - Example: TCSP
Operational Semantics: Syntax

Syntax: Process terms over alphabet $A$

- Blocking process: $0$
- Action prefix for some $a$ or $\tau$ of $A$: $a.P$
- Alternative choice: $P + Q$
- Parallel composition: $P | Q$
- (Mutual) Recursion: $P = Q$

\[
S = p.x.S \\
R = x.c.R \\
E = p.\tau.B \\
B = c.E + p.F \\
F = c.\tau.B \\
P = S | R
\]
Operational semantics:

- Defined by (labeled) transition relation
- Constructed in terms of computation steps
  - Start state
  - (Inter)action
  - End state
- Structured operational semantics: Process terms as state space
Example: Constructing execution steps

- Construct tableau of rules
- Use structure of process terms
  - Break down terms into atomic terms
  - Build up tableau using rules
Operational Semantics: Verification

\[ \approx \text{ is a (strong) bisimulation } \iff \forall P, Q, a. \]

\[
P \approx Q \Rightarrow (\forall P'. P \xrightarrow{a} P' \Rightarrow \exists Q'. Q \xrightarrow{a} Q' \land P' \approx Q') \land \\
(\forall Q'. Q \xrightarrow{a} Q' \Rightarrow \exists P'. P \xrightarrow{a} P' \land P' \approx Q')
\]

Verification:
- Relation between process terms \( S_1 \) and \( S_2 \)
- Based on transition relation:
  - Strong Bisimilarity: \( \exists \approx \cdot S_1 \approx S_2 \)
  - Weak bisimilarity: Bisimilarity modulo \( \tau \)
  - Simulation relation: Asymmetric variant
Operational Semantics: Example

Example: Proving process bisimilarity of process terms $S_1$ and $S_2$

- **Approach:**
  - Construct largest bisimulation relation over transition relation
  - Check containment of process terms $S_1$ and $S_2$

- **Automatic construction:**
  - Using fixed point induction
  - Using 1-step-equivalence as induction principle
Models

Classes of models:

• **Algebraic model**: Specification in form of syntactic formulae
  – Concepts: Process
  – Verification: Term equivalence
  – Example: ACP

• **Operational model**: Specification in form of abstract machine
  – Concepts: State, (labeled) transition
  – Verification: (Bi-)Simulation
  – Example: CCS

• **Denotational model**: Specification in form of observable behavior
  – Concepts: Event, (observation) trace
  – Verification: Behavior Inclusion
  – Example: TCSP
Denotational Semantics: Syntax

\[
S = \mu X : p.x.X \\
R = \mu X : x.c.X \\
E = p.B \\
B = \mu X : c.p.X + p.c.X \\
P = S \mid R
\]

Syntax: Process terms over alphabet \( A \)

- Blocking process: \( 0 \)
- Action prefix for some \( a \) of \( A \): \( a.P \)
- Alternative choice: \( P + Q \)
- Parallel composition: \( P \mid Q \)
- (Mutual) Recursion: \( \mu P : T(P) \)
Denotational Semantics: Mapping

\[
F[0] = \{(\langle\rangle, R) \mid R \subseteq A\}
\]
\[
F[a.P] = \{(\langle\rangle, R) \mid R \subseteq A \setminus \{a\}\} \cup \{(a \cdot t, R) \mid (t, R) \in F[P]\}
\]
\[
F[P + Q] = \{(t, R) \mid ((t, R) \in F[P] \cap F[Q]) \lor (t \neq \langle\rangle \land (t, R) \in F[P] \cup F[Q])\}
\]
\[
F[P \mid Q] = \{(t, R_1 \cup R_2) \mid (t \uparrow \alpha P, R_1) \in F[P] \land (t \uparrow \alpha Q, R_2) \in F[Q]\}
\]
\[
F[\mu P : T(P)] = \text{lfp}_i F[T^i](A^* \times 2^A)
\]

Denotational semantics:

- Defined by mapping into (Scott) domain
- Constructed via inductive definition
  - Compositional
  - Aligned according to process term structure
- Recursive definitions: Using fixed point induction
Denotational Semantics: Example

\[ F[S] = \{(\leftrightarrow, \emptyset), (\leftrightarrow, \{x\}), (p, \emptyset), (p, \{p\}), (p \cdot x, \emptyset), (p \cdot x, \{x\}), \ldots\} \]

\[ F[R] = \{(\leftrightarrow, \emptyset), (\leftrightarrow, \{c\}), (x, \emptyset), (x, \{x\}), (x \cdot c, \emptyset), (x \cdot c, \{c\}), \ldots\} \]

\[ F[S \mid R] = \{(\leftrightarrow, \emptyset), (\leftrightarrow, \{c\}), (\leftrightarrow, \{x\}), (\leftrightarrow, \{c, x\}),
(p, \emptyset), (p, \{p\}), (p, \{c\}), (p, \{p, c\}),
(p \cdot x, \emptyset), (p \cdot x, \{x\}), \ldots\} \]

Example: Constructing behaviors

- Construct behaviors as elements of Scott domain
- Use structure of process terms
  - Break down terms into atomic terms
  - Build up behaviors using rules
  - Use fixed point induction
Operational Semantics: Verification

Verification:
• Relation between process terms $S_1$ and $S_2$
• Based on denotation of process terms:
  – Equivalence: $P \leq Q \land Q \leq P$
  – Refinement: Ordering in Scott domain

\[ P \leq Q \iff F[P] \subseteq F[Q] \]
Operational Semantics: Example

\[ S = \mu X : p.x.X \]
\[ R = \mu X : x.c.X \]
\[ E = p.B \]
\[ B = \mu X : c.p.X + p.c.X \]
\[ P = S | R \]

\[ F[S | R] = \{ (\langle \rangle, \emptyset), (\langle \rangle, \{ c \}), (\langle \rangle, \{ x \}), (\langle \rangle, \{ c, x \}), \]
\[ (p, \emptyset), (p, \{ p \}), (p, \{ c \}), (p, \{ p, c \}), \]
\[ (p \cdot x, \emptyset), (p \cdot x, \{ x \}), ... \} \]

= \[ F[E] \]

Example: Proving equivalence of process terms \( S_1 \) and \( S_2 \)

- Approach:
  - Construct denotation of process terms \( S_1 \) and \( S_2 \)
  - Check set equivalence
Conclusion

• Algebraic Semantics
  – Advantage: Semantics is term language
  – Disadvantages: Consistency, implementability

• Operational Semantics:
  – Advantages: Implementability
  – Disadvantages: Verification reduced to simulation

• Denotational Semantics:
  – Advantages: Observational approach
  – Disadvantages: Complex