REASONING ABOUT PARALLEL PROGRAMS WITH LOCAL VARIABLES

Stephen Brookes
Carnegie Mellon University
School of Computer Science

MFPS’94
SHARED VARIABLE PARALLELISM

• Parallel imperative programs reading and writing shared memory

• $C_1 || C_2$ modelled by interleaving of atomic actions

• Synchronization using conditional atomic action: \texttt{await \ensuremath{B} then} $C$

Coarseness Assumption
Assignment and boolean expressions are atomic.
OPERATIONAL SEMANTICS

\[
\begin{align*}
\langle C_1, s \rangle & \rightarrow \langle C'_1, s' \rangle \\
\langle C_1 \parallel C_2, s \rangle & \rightarrow \langle C'_1 \parallel C_2, s' \rangle \\
\langle C_2, s \rangle & \rightarrow \langle C'_2, s' \rangle \\
\langle C_1 \parallel C_2, s \rangle & \rightarrow \langle C_1 \parallel C'_2, s' \rangle \\
\langle C_1, s \rangle \text{ term} \quad \langle C_2, s \rangle \text{ term} & \quad \langle C_1 \parallel C_2, s \rangle \text{ term} \\
\langle B, s \rangle \rightarrow^* \texttt{tt} \quad \langle C, s \rangle \rightarrow^* \langle C', s' \rangle \text{ term} & \quad \langle \text{await } B \text{ then } C, s \rangle \rightarrow \langle \text{skip}, s' \rangle
\end{align*}
\]

cf. Hennessy and Plotkin (1979)
PROGRAM BEHAVIOR

• Partial correctness:
\[ \mathcal{M}[C] = \{(s, s') \mid \langle C, s \rangle \rightarrow^* \langle C', s' \rangle \text{term}\} \]

• Strong correctness:
\[ \mathcal{M}[C] = \{(s, s') \mid \langle C, s \rangle \rightarrow^* \langle C', s' \rangle \text{term}\}
\cup \{(s, \bot) \mid \langle C, s \rangle \rightarrow^\omega\} \]

• Total correctness:
\[ \mathcal{M}[C] = \{(s, s') \mid \langle C, s \rangle \rightarrow^* \langle C', s' \rangle \text{term}\}
\cup \{(s, s') \mid s' \in S_\bot \land \langle C, s \rangle \rightarrow^\omega\} \]

• Deadlock:
\[ \mathcal{M}[C] = \{(s, s') \mid \langle C, s \rangle \rightarrow^* \langle C', s' \rangle \text{dead}\} \]
FULL ABSTRACTION

Milner (1977):
A semantics is *fully abstract* if two phrases have the same meaning precisely when they induce the same behavior in all program contexts.

• A natural criterion for judging merit of a semantics.
• Often difficult to achieve.
• A fully abstract semantics supports compositional reasoning.
• Failure of full abstraction may suggest:
  – defective semantic model
  – missing language features
TRACE SEMANTICS

Transitions: $\Sigma = S \times S$
Transition traces: $\Sigma^\infty = \Sigma^+ \cup \Sigma^\omega$
Trace semantics: $T[C] \subseteq \Sigma^\infty$

- A trace represents a sequence of snapshots taken during computation, allowing for possible interruption.

- Operational definition:

$$T[C] = \{(s_0, s'_0)(s_1, s'_1) \ldots (s_k, s'_k) \mid \langle C, s_0 \rangle \rightarrow^* \langle C_1, s'_0 \rangle \& \langle C_1, s_1 \rangle \rightarrow^* \langle C_2, s'_1 \rangle \& \ldots \ldots \ldots \& \langle C_k, s_k \rangle \rightarrow^* \langle C', s'_k \rangle \text{term}\}$$

- Partial correctness behavior corresponds to “interference-free” subset:

$$M[C] = \{(s, s') \mid (s, s') \in T[C]\}.$$
PROPERTIES

• $\mathcal{T}[C]$ is closed under stuttering:
  $$\alpha\beta \in \mathcal{T}[C] \Rightarrow \alpha(s, s)\beta \in \mathcal{T}[C].$$

• $\mathcal{T}[C]$ is closed under mumbling:
  $$\alpha(s, s')(s', s'')\beta \in \mathcal{T}[C] \Rightarrow \alpha(s, s'')\beta \in \mathcal{T}[C].$$

DEFINITION

For $T \subseteq (S \times S)^+$ let $T^\dagger$ be smallest closed set including $T$.

FACT

Closed sets of traces, ordered by inclusion, form a complete lattice.
DENOTATIONAL SEMANTICS

\[ T[\text{skip}] = \{(s, s) \mid s \in S\} \]

\[ T[I:=E] = \{(s, [s|I = n]) \mid (s, n) \in \mathcal{E}[E]\} \]

\[ T[C_1; C_2] = T[C_1]; T[C_2] \]
\[ = \{\alpha\beta \mid \alpha \in T[C_1] \& \beta \in T[C_2]\} \]

\[ T[C_1||C_2] = T[C_1]; T[C_2] \]
\[ = \bigcup\{\alpha||\beta \mid \alpha \in T[C_1] \& \beta \in T[C_2]\} \]

\[ T[\text{await} B \text{ then } C] = \]
\[ \{(s, s') \in T[C] \mid (s, \text{tt}) \in B[B]\} \]

\[ T[\text{if} B \text{ then } C_1 \text{ else } C_2] = \]
\[ T[B]; T[C_1] \cup T[\neg B]; T[C_2] \]

\[ T[\text{while} B \text{ do } C] = (T[B]; T[C])^*; T[\neg B] \]

where \( T[B] = \{(s, s) \mid (s, \text{tt}) \in B[B]\} \).
PROPERTIES

• Sequential and parallel composition are continuous operations on closed sets of traces.

• Loop semantics can be expressed as a least fixed point:

\[ \mathcal{T}[\textbf{while } B \textbf{ do } C] = \mu T.(\mathcal{T}[B]; \mathcal{T}[C]; T \cup \mathcal{T}[\neg B]) \]

• Denotational and operational definitions of \( \mathcal{T} \) coincide.
INFINITE TRACES

• Let $\Sigma$ be $S \times S$.

• $T[C] \subseteq \Sigma^\infty = \Sigma^* \cup \Sigma^\omega$

• $T[C]$ is closed under stuttering and mumbling.

• Extend concatenation to infinite traces:
  
  $\alpha\beta = \alpha$ if $\alpha$ is infinite.

• Loop semantics as an “operational” fixed point:
  
  $T[\text{while } B \text{ do } C] =$
  
  $(T[B]; T[C])^*; T[\neg B]$
  
  $\cup (T[B]; T[C])^\omega$

• Full abstraction for strong correctness.
FAIR PARALLELISM

“No parallel component is delayed forever”

- \( \mathcal{T}[C_1 \parallel C_2] = \mathcal{T}[C_1] \parallel \mathcal{T}[C_2] \)
- \( T_1 \parallel T_2 = \cup \{ \alpha \parallel \beta \mid \alpha \in T_1 \land \beta \in T_2 \} \)
- \( \alpha \parallel \beta = \{ \gamma \mid (\alpha, \beta, \gamma) \in \text{fairmerge} \} \)
- \( \text{fairmerge} = (L^*RR^*L)^\omega \cup (L \cup R)^*A \), where
  \[
  L = \{ (\sigma, \epsilon, \sigma) \mid \sigma \in \Sigma \} \\
  R = \{ (\epsilon, \sigma, \sigma) \mid \sigma \in \Sigma \} \\
  A = \{ (\epsilon, \alpha, \alpha), (\alpha, \epsilon, \alpha) \mid \alpha \in \Sigma^* \cup \Sigma^\omega \} 
  \]
  cf. Park (1979)
- Fair parallel composition is a \textit{continuous function} on closed sets of traces.
RELATED WORK

• Abrahamson (1979), Park (1979):
  – no stuttering or mumbling
  – not fully abstract

• de Boer, Kok, Palamidessi, Rutten (1991):
  – only restricted mumbling
  – different language and behavior
  – ignores fairness

• Abadi, Plotkin (1993):
  – finite traces, stuttering, mumbling
  – closed under prefix
  – full abstraction for safety properties
  – ignores fairness
FURTHER RESEARCH

• Connection with logic:
  – generalized Hoare logics
  – safety and liveness
  – modal logics (e.g. LTL, CTL, $\mu$-calculus)

• Laws of program equivalence, for:
  – program transformation, derivation
  – parallelization
  – simplification of program proofs

• Full abstraction for fairly communicating processes
  – Hoare’s CSP
  – Milner’s CCS with value-passing