Fairness, resources, and separation

Stephen Brookes

Cambridge Concurrency Workshop
July 2010
Fairness

● Key ingredient in semantics
  ● parallel composition = *fair interleaving*
  ● “any reasonable scheduler is *fair*”

● Crucial for *liveness* analysis
  ● need to exclude irrelevant *unfair* executions

● Technical problems
  ● lack of monotonicity
  ● unbounded non-determinism
Park’s fairmerge

shared-memory parallel programs

• Program denotes set of traces over $\Sigma = S \times S$

• $\llbracket c_1; c_2 \rrbracket = \{ \alpha_1 \alpha_2 \mid \alpha_1 \in \llbracket c_1 \rrbracket, \alpha_2 \in \llbracket c_2 \rrbracket \}$

• $\llbracket c_1 || c_2 \rrbracket = \{ \alpha \mid \exists \alpha_1 \in \llbracket c_1 \rrbracket, \alpha_2 \in \llbracket c_2 \rrbracket.
\quad (\alpha_1, \alpha_2, \alpha) \in fairmerge \}$

$\alpha$ is a fair merge of $\alpha_1$ and $\alpha_2$

Park 1979
**fairmerge** as a fixed point

... but not least or greatest

\[
F : \mathcal{P}(\Sigma^{\infty} \times \Sigma^{\infty} \times \Sigma^{\infty}) \rightarrow \mathcal{P}(\Sigma^{\infty} \times \Sigma^{\infty} \times \Sigma^{\infty})
\]

\[
F(R) = \text{copy} \cup \text{left } R \cup \text{right } R
\]

\[
\text{copy} = \{(\varepsilon, \alpha, \alpha) | \alpha \in \Sigma^{\infty}\} \cup \{(\alpha, \varepsilon, \alpha) | \alpha \in \Sigma^{\infty}\}
\]

\[
\text{left} = \{(\lambda, \varepsilon, \lambda) | \lambda \in \Sigma\} \quad \text{right} = \{(\varepsilon, \lambda, \lambda) | \lambda \in \Sigma\}
\]

\[
F(\text{fairmerge}) = \text{fairmerge}
\]

But \( \mu F \subset \text{fairmerge} \subset \nu F \)

*only finite cases*

contains unfair merges, e.g. \((0^w, 1^w, 0^w)\)
... as an iterated fixed point

\[ F : \mathcal{P}(\Sigma^\infty \times \Sigma^\infty \times \Sigma^\infty)^2 \rightarrow \mathcal{P}(\Sigma^\infty \times \Sigma^\infty \times \Sigma^\infty) \]
\[ F(X,Y) = \text{copy} \cup \text{left } X \cup \text{right } Y \]

\[ \text{fairmerge} = F(\text{fairmerge}, \text{fairmerge}) \]
\[ \text{fairmerge} = \nu X. \mu Z. F(Z, \mu Y. F(X,Y)) \]

nested combination of

greatest and least fixed points

Hence \[ \text{fairmerge} = (\text{left} \cup \text{right})^* \text{copy} \cup (\text{left}^* \text{right} \text{right}^* \text{left})^\omega \]

iteration formula
Associativity

$$[c_1 || (c_2 || c_3)] = [(c_1 || c_2) || c_3]$$

- Semantics should validate natural laws
- But Park’s formulation of *fairmerge* does not permit a simple proof
  - *fixed point definition* complicated by nesting of $\mu, \nu$
  - *iteration formula* hard to generalize

“... other algebraic insights seem to be called for...”

Park 1981
A recurring theme

- Dataflow networks
  - fairmerge as a process

- Shared memory
  - fairness + mutual exclusion

- Concurrent separation logic
  - fairness + resources & race detection

... all involve some form of fairmerge
Contributions

- A simple yet general \textit{fairmerge}
  - straightforward proofs of algebraic properties

- Can handle \textit{resources} and \textit{race-detection}
  - based on abstract model of resources and conflict

- Relationship with earlier work
  - Park’s \textit{fairmerge}
  - \textit{concurrent separation logic}
  - trace semantics for \textit{dataflow}
Prefixes

\[ pre : \Sigma^\infty \rightarrow \mathcal{P}(\Sigma^*) \]

- \( pre(\alpha) = \{ \beta \in \Sigma^* \mid \beta \leq \alpha \} \)
- \( (pre(\alpha), <) \) is a linear order

Strictly monotone

\[ f : pre(\alpha) \rightarrow pre(\beta) \text{ is strictly monotone iff} \]

- \( f(\varepsilon) = \varepsilon \)
- \( \alpha_1 < \alpha_2 \) implies \( f(\alpha_1) < f(\alpha_2) \)
**k-ary fairmerge**

\[ \text{FM}_k \subseteq (\Sigma^\infty)^k \times \Sigma^\infty \]

\(((\alpha_1, ..., \alpha_k), \beta) \in \text{FM}_k \text{ iff } \exists \text{ strictly monotone functions } \]

\[ f_i : \text{pre}(\alpha_i) \to \text{pre}(\beta) \text{ such that } \]

- \( \text{rge}(f_1) \cup ... \cup \text{rge}(f_k) = \text{pre}(\beta) \)
- \( \text{rge}(f_i) \cap \text{rge}(f_j) = \{\varepsilon\} \text{ for } i \neq j \)
- \( \text{last}(f_i(\alpha)) = \text{last}(\alpha) \text{ for } \alpha \in \text{pre}(\alpha_i) \)

(f_1, ..., f_k) is a “schedule” for merging \( \alpha_1, ..., \alpha_k \) to \( \beta \)
Facts

- $\text{FM}_1 = \text{identity relation}$
- $\text{FM}_2 = \text{Park's fairmerge}$
- Can use $\text{FM}_3$ to prove fairmerge associative

\[
((\alpha_1, \alpha_2, \alpha_3), \gamma) \in \text{FM}_3
\quad \text{if and only if}
\exists \beta. ((\alpha_1, \alpha_2), \beta) \in \text{FM}_2 \land ((\alpha_3, \beta), \gamma) \in \text{FM}_2
\]

\[
(\alpha_1 \| \alpha_2) \| \alpha_3 = \alpha_1 \| (\alpha_2 \| \alpha_3) = \alpha_1 \| \alpha_2 \| \alpha_3
\]
nodes are relations,
connection = composition
General associativity

Proof: by schedule construction
Properties

• Invariance under permutation

If \(((\alpha_1, ..., \alpha_k), \beta) \in FM_k\)
then \(((\alpha_{\pi 1}, ..., \alpha_{\pi k}), \beta) \in FM_k\)

• Invariance under map

If \(((\alpha_1, ..., \alpha_k), \beta) \in FM_k\) and \(h: \Sigma \rightarrow \Sigma\)
then \(((map \ h \ \alpha_1, ..., \ map \ h \ \alpha_k), \ map \ h \ \beta) \in FM_k\)
Prefix / suffix

• If \(((\alpha_1, \ldots, \alpha_k), \beta) \in \text{FM}_k\) and \(\beta = \beta'\beta''\) then there are traces \(\alpha_i', \alpha_i''\) such that \(\alpha_i = \alpha_i'\alpha_i''\) and \((\left(\alpha_1', \ldots, \alpha_k'\right), \beta') \in \text{FM}_k\) and \((\left(\alpha_1'', \ldots, \alpha_k''\right), \beta'') \in \text{FM}_k\)

Concatenation

• If \(((\alpha_1', \ldots, \alpha_k'), \beta') \in \text{FM}_k, \beta' \in \Sigma^*,\) and \((\left(\alpha_1'', \ldots, \alpha_k''\right), \beta'') \in \text{FM}_k\) then \((\left(\alpha_1'\alpha_1'', \ldots, \alpha_k'\alpha_k''\right), \beta'\beta'') \in \text{FM}_k\)

Proofs: by schedule composition and decomposition

not true if \(\beta'\) is infinite, because of non-monotonicity
Fairness + resources

Extend framework to handle synchronization

• Use abstract model of resources and enabling
• Interleaving must maintain resource compatibility
  • mutual exclusion
  • disjointness
• Enabling of actions is sensitive to resources
Resource model

A set $M$ of resource values

A partial composition $\oplus : M \times M \rightarrow M$

$m_1, m_2$ compatible iff $m_1 \oplus m_2$ defined

A zero element $0 \in M$

- $m \oplus 0 = m$
- $m_1 \oplus m_2 = m_2 \oplus m_1$
- $(m_1 \oplus m_2) \oplus m_3 = m_1 \oplus (m_2 \oplus m_3)$
- $m \oplus m_1 = m \oplus m_2$ implies $m_1 = m_2$

O’Hearn, Yang, Calcagno

Tuesday, July 6, 2010
Examples

• Trivial
  \[ M = \{0\} \]

• Resource names
  \[ (\mathcal{P}_{\text{fin}}(R), \cup, \{\}) \]
  A, B compatible iff \( A \cap B = \{\} \)

• Heaps
  \[ (H, \cup, \{\}) \quad H = \text{Loc} \rightarrow_{\text{fin}} V \]
  h, h’ compatible iff \( \text{dom}(h) \cap \text{dom}(h’) = \{\} \)

  also Stores, Permissions, ...
Enabling

\[ m_2 \leftarrow m_1 \xrightarrow{\lambda} m_1' \]

*Process* with resources \( m_1 \), can do \( \lambda \), in *environment* with resources \( m_2 \)

- Resources must be *compatible*
- Ability to do \( \lambda \) depends on \( m_1, m_2 \) and may *change* the resources of process
- *Frame* properties describe the effect of “extra” resources
Frame properties

\[ m_2 \vdash m_1 \xrightarrow{\lambda} m_1' \quad \text{implies} \quad m' = m \oplus m_1' \]

\[ m_2 \vdash m_1 \xrightarrow{\lambda} m_1' \quad \text{implies} \quad m' = m_1' \]

\[ m \oplus m_2 \vdash m_1 \xrightarrow{\lambda} m_1' \quad \text{implies} \quad m_2 \vdash m_1 \xrightarrow{\lambda} m_1' \]

extra process resources propagate
extra environment resources don’t affect process
removing environment resources doesn’t disable process

subject to compatibility constraints
Examples

• Trivial

\[ 0 \vdash 0 \xrightarrow{\lambda} 0 \quad \text{for all } \lambda \]

• Resource names

\[ \Sigma = \{\text{acq } r, \text{try } r, \text{rel } r \mid r \in R\} \cup \Delta \]

\[ A, B \in M = \mathbb{P}_{\text{fin}}(R), \quad A \cap B = \{\} \]

<table>
<thead>
<tr>
<th>Action</th>
<th>Transition</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{acq } r</td>
<td>( B \vdash A \xrightarrow{\text{acq } r} A \cup {r} )</td>
<td>if ( r \notin A \cup B )</td>
</tr>
<tr>
<td>\text{try } r</td>
<td>( B \vdash A \xrightarrow{\text{try } r} A )</td>
<td>if ( r \in A \cup B )</td>
</tr>
<tr>
<td>\text{rel } r</td>
<td>( B \vdash A \xrightarrow{\text{rel } r} A - {r} )</td>
<td>if ( r \in A )</td>
</tr>
<tr>
<td>\text{try } \lambda</td>
<td>( B \vdash A \xrightarrow{\lambda} A )</td>
<td>if ( \lambda \in \Delta )</td>
</tr>
</tbody>
</table>

*enabling enforces mutual exclusion*
Fairness + resources

- Extend $FM_k$ with resources $M$ and enabling $\vdash$

$$FM_k \in \mathcal{P}((\Sigma^\infty)^k \times \Sigma^\infty)$$

$$\Leftrightarrow$$

$$RFM_k : M^k \to \mathcal{P}((\Sigma^\infty)^k \times \Sigma^\infty)$$

$$((\alpha_1, ..., \alpha_k), \beta) \in RFM_k(m_1, ..., m_k)$$

iff $\beta$ is a resource-sensitive fair merge of $\alpha_1, ..., \alpha_k$ starting from $m_1, ..., m_k$
\( \text{RFM}_k \)

\(((\alpha_1, \ldots, \alpha_k), \beta) \in \text{RFM}_k(m_1, \ldots, m_k) \)

iff \( \exists \) schedule \( f_i : \text{pre}(\alpha_i) \rightarrow \text{pre}(\beta) \) as before, and a function \( \text{res} : \text{pre}(\beta) \rightarrow M^k \) such that

- \( \text{res}(\varepsilon) = (m_1, \ldots, m_k) \)
- If \( f_i(\alpha') = \beta'\lambda, \)
  \( \text{res}(\beta') = (m_1', \ldots, m_k'), \)
  & \( \text{res}(\beta'\lambda) = (m_1'', \ldots, m_k''), \)
then
  \( (\oplus_{j \neq i} m_j') \vdash m_i' \overset{\lambda}{\rightarrow} m_i'' \)
and \( m_j'' = m_j' \) for \( j \neq i \)
RFM$_2$

For finite traces...  
all resource-sensitive interleavings

$$(\lambda \alpha) \parallel (\mu \beta) =$$

$$m_1 m_2 \backslash \{ \lambda \gamma \mid m_2 \vdash m_1 \xrightarrow{\lambda} m_1' \land \gamma \in \alpha \parallel (\mu \beta) \}$$

$$\cup \{ \mu \gamma \mid m_1 \vdash m_2 \xrightarrow{\mu} m_2' \land \gamma \in (\lambda \alpha) \parallel \beta \}$$

$$\varepsilon \parallel \beta = \{ \beta \mid m_1 \vdash m_2 \xrightarrow{\beta} m_2' \}$$

$$\alpha \parallel \varepsilon = \{ \alpha \mid m_2 \vdash m_1 \xrightarrow{\alpha} m_1' \}$$
Properties

Resource-sensitive analogs of

- Prefix/suffix
- Concatenation
- General associativity

\[
\alpha \ || \ \frac{(\beta \ || \ \gamma) = (\alpha \ || \ \beta) \ || \ \gamma}{m_1 \ m_2 \oplus m_3 \ m_2 \ m_3 \ m_2 \ m_1 \oplus m_2 \ m_3}
\]

Proofs: as before, but adapted for resources
Example

\[ \Sigma = \{ acq \ r, \ try \ r, \ rel \ r \mid r \in R \} \cup \Delta \]

\[ M = (P_{\text{fin}}(R), \cup, \{ \}) \]

- \[ [\textbf{with} \ r \ \textbf{when} \ b \ \textbf{do} \ c] = wait^* \ \textbf{enter} \cup \textbf{wait}^\omega \]

  \[ \text{wait} = (acq \ r) \ [b]_{\text{false}} \ (rel \ r) \cup \{\text{try} \ r\} \]

  \[ \text{enter} = (acq \ r) \ [b]_{\text{true}} \ [c] \ (rel \ r) \]

- \[ [c_1 || c_2] = \{ \alpha \mid \exists \alpha_1 \in [c_1], \alpha_2 \in [c_2] \}

  \[ ((\alpha_1, \alpha_2), \alpha) \in \text{RFM}_2(\{ \}, \{ \}) \} \]

  \[ \alpha \upharpoonright \{acq \ r, rel \ r\} \leq (acq \ r \ rel \ r)^\omega \]

  \[ \text{for all } r \]

\[ \text{mutual exclusion} \]

\[ \text{for each resource} \]

Tuesday, July 6, 2010
Fairness + conflict

- Concurrent actions may interfere, e.g. race conditions
- Extend $\text{RFM}_k(M, \vdash)$ with a conflict relation
  - treat conflict as a disaster
  - will support proofs of conflict-freedom

\[
\begin{align*}
\triangleleft \text{RFM}_k : M^k & \rightarrow \mathcal{P}((\Sigma^\infty)^k \times \Sigma^\infty) \\
\triangleright \text{RFM}^\wedge_k : M^k & \rightarrow \mathcal{P}((\hat{\Sigma}^\infty)^k \times \hat{\Sigma}^\infty)
\end{align*}
\]

\[
((\alpha_1, \ldots, \alpha_k), \beta) \in \text{RFM}_k(m_1, \ldots, m_k)
\]
iff $\beta$ is a conflict-detecting fair merge of $\alpha_1, \ldots, \alpha_k$ starting from $m_1, \ldots, m_k$
Conflict

- **Actions**
  \[ \hat{\Sigma} = \Sigma \cup \{ \text{abort} \} \]

- **Conflict relation**
  \[ \bowtie \subseteq \hat{\Sigma} \times \hat{\Sigma} \]

- **symmetric**
  \[ \lambda \bowtie \mu \iff \mu \bowtie \lambda \]

- **strict**
  \[ \lambda \bowtie \text{abort} \quad \text{iff} \quad \text{abort} \bowtie \lambda \]

*a symmetric binary relation on \( \Sigma \), extended strictly to \( \hat{\Sigma} \)*
Examples

• Concurrent separation logic
  \( \lambda \Join \mu \) if \( \text{dom}(\lambda) \cap \text{dom}(\mu) \neq \{ \} \)
  \( \text{heap actions on same cell} \)

• Permissions
  \( \lambda \Join \mu \) if \( \text{writes}(\lambda) \cap \text{dom}(\mu) \neq \{ \} \)
  or \( \text{writes}(\mu) \cap \text{dom}(\lambda) \neq \{ \} \)
  \( \text{concurrent write to same variable} \)

\( \text{conflict} = \text{race condition} \)
$\text{RFM}_k$

$((\alpha_1, \ldots, \alpha_k), \beta) \in \text{RFM}_k(\bar{m})$

iff

$((\alpha_1, \ldots, \alpha_k), \beta) \in \text{RFM}_k(\bar{m})$

or

$\beta = \beta' \text{ abort } \& \exists i \neq j. \exists \lambda, \mu \in \sum$

$\exists \alpha_1' \in \text{pre}(\alpha_1), \ldots, \alpha_k' \in \text{pre}(\alpha_k),$

$((\alpha_1', \ldots, \alpha_k'), \beta') \in \text{RFM}_k(\bar{m})$

$\& \alpha_i' \lambda \in \text{pre}(\alpha_i) \& \alpha_j' \mu \in \text{pre}(\alpha_j) \& \lambda \bowtie \mu$

resource-sensitive fair merges, as before

disaster if schedule reaches a conflict
RFM$_2$

For finite traces... *all conflict-detecting, resource-sensitive interleavings*

\[
(\lambda\alpha) \parallel (\mu\beta) = \\
\begin{array}{ll}
\{ \lambda\gamma \mid m_2 \vdash m_1 \xrightarrow{\lambda} m_1' \quad & \gamma \in \alpha \parallel (\mu\beta) \}
\end{array}
\]

\[
\bigcup \{ \mu\gamma \mid m_1 \vdash m_2 \xrightarrow{\mu} m_2' \quad & \gamma \in (\lambda\alpha) \parallel \beta, \}
\]

\[
\bigcup \{ \text{abort} \mid \lambda \bowtie \mu \}
\]

A semantics for concurrent separation logic, Brookes04
Dataflow

• Network of non-deterministic processes
• Communication on named channels
• Process denotes trace set over $((I \cup O) \times V)^\infty$
• k-ary fairmerge process
  \[ F({i_1, \ldots, i_k}, \{o\}) \]
  with traces characterized via mapping from output occurrences to input channel indices

Shangbhogue 1990
**Connection Theorem**

\[
F(\{i_1, ..., i_k\}, \{o\}) = \text{buf}(i_1, o) \parallel ... \parallel \text{buf}(i_k, o)
\]

where \(\alpha \in \text{buf}(i, o)\) iff

\[
data(\alpha \uparrow o) = data(\alpha \uparrow i) \land \forall \beta \leq \alpha. \ data(\beta \uparrow o) \leq data(\beta \uparrow i)
\]

\[
F(\{i_1, ..., i_k\}, \{o\}) = \{ \alpha \mid \exists \alpha_1 \in \text{buf}(i_1, o), ..., \alpha_k \in \text{buf}(i_k, o). ((\alpha_1, ..., \alpha_k), \alpha) \in FM_k \}
\]

\[
FM_k = \{ (((data(\alpha \uparrow i_1), ..., data(\alpha \uparrow i_k)), data(\alpha \uparrow o)) \mid \alpha \in F(\{i_1, ..., i_k\}, \{o\}) \}
\]
Composition Theorems

• Shanbhogue:
  a dataflow network composed of fairmerge processes is trace-equivalent to a single fairmerge

• Here:
  a composition of fairmerge relations is equal to a single fairmerge

Similar in form, but ours not limited to dataflow
Results are equipollent, by Connection Theorem
Not monotone

- If $(\alpha, \beta, \gamma) \in \text{fairmerge}$ and $\alpha \leq \alpha'$, $\beta \leq \beta'$
  there may be no $\gamma'$ such that
  $\gamma \leq \gamma'$ and $(\alpha', \beta', \gamma') \in \text{fairmerge}$

... not a problem
Conclusions

• A general *compositional* account of *fairness*
  • resources, enabling, conflict

• Rational reconstruction of earlier models
  • Park’s fairmerge, Owicki-Gries logic, concurrent separation logic, dataflow semantics

• Future plans
  • Communicating processes, channel fairness
  • Actions with axioms, e.g. stuttering, mumbling
  • Other algebraic properties