Modeling Task Systems Using Parameterized Partial Orders

Fred Houben\textsuperscript{1} Georgeta Igna\textsuperscript{2} Frits Vaandrager\textsuperscript{2}

\textsuperscript{1}ASML

\textsuperscript{2}ICIS, Radboud Universiteit Nijmegen, the Netherlands

RTAS, Beijing, April 2012
Datapath Scheduling for Printers
Design Space Exploration Using Y-Chart Pattern

Fred Houben, Georgeta Igna and Frits Vaandrager

Modeling Task Systems Using Parameterized Partial Orders
Example of Y-Chart Pattern

Requirements:
- maximize throughput
- minimize $M_1 + M_2$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>1</td>
<td>2</td>
<td>20%</td>
</tr>
<tr>
<td>normal</td>
<td>3</td>
<td>6</td>
<td>50%</td>
</tr>
<tr>
<td>high</td>
<td>10</td>
<td>20</td>
<td>30%</td>
</tr>
</tbody>
</table>
Outcome Octopus Project

Toolset for design space exploration of real-time embedded systems

See http://dse.esi.nl/.
Analysis Tools

- **CPN tools** (simulation)
- **Uppaal** (reachability analysis)
  - for jobs with known arrival times, what is the schedule that achieves the best latency/makespan?
  - for jobs with unknown arrival times and for a fixed scheduler, what is the worst case latency/makespan?
- **Uppaal Tiga** (timed games)
  - for jobs with unknown arrival times, what is the scheduler that achieves best latency/makespan?
- **SDF toolset** (bounds on worst case latency/makespan)
- ...
Example of Analysis with Uppaal Tiga

Fred Houben, Georgeta Igna and Frits Vaandrager

Modeling Task Systems Using Parameterized Partial Orders
Finite Repetitive Behavior

Observation: task graphs often contain finite repetitive behavior
Finite Repetitive Behavior

Observation: task graphs often contain finite repetitive behavior

- order beer brewery consists of several pallets, each containing several crates, each containing several bottles
Finite Repetitive Behavior

Observation: task graphs often contain finite repetitive behavior

- order beer brewery consists of several pallets, each containing several crates, each containing several bottles
- order for semiconductor industry consists of several lots, each containing several wafers, each containing multiple masks
Observation: task graphs often contain finite repetitive behavior

- order beer brewery consists of several pallets, each containing several crates, each containing several bottles
- order for semiconductor industry consists of several lots, each containing several wafers, each containing multiple masks
- order for printer consists of number of copies of a file, each file consisting of several pages
Finite Repetitive Behavior

Observation: task graphs often contain finite repetitive behavior

- order beer brewery consists of several pallets, each containing several crates, each containing several bottles
- order for semiconductor industry consists of several lots, each containing several wafers, each containing multiple masks
- order for printer consists of number of copies of a file, each file consisting of several pages
- ...
Representing Task Graphs with Finite Repetitive Behavior

- Colored Petri Nets
- UML activity diagrams ← Used by ASML engineers
- Parametrized Partial Orders ← Used by Océ engineers
- ...
PPO for Printer Use Case

Fred Houben, Georgeta Igna and Frits Vaandrager

Modeling Task Systems Using Parameterized Partial Orders
PPO for Wafer Production Use Case

\[ l' = l + 1 \]

\[ (w < 15 \land l' = l \land w' = w + 1) \lor \]
\[ (w = 15 \land l' = l + 1 \land w' = 1) \]
A parameterized partial order is a tuple \( A = (\mathcal{T}, \mathcal{M}, E, U) \) where

- \( \mathcal{T} \) is a finite set of tasks
- \( \mathcal{M} \) assigns to each task \( T \) a finite set of parameters; \( V(T) \) denotes the set of valuations of parameters in \( \mathcal{M}(T) \)
- \( E \subseteq \{s, e\} \times \mathcal{T} \times \{s, e\} \times \mathcal{T} \) is a set of edges; we require, for each \( T \in \mathcal{T} \), \( (s, T, e, T) \in E \)
- For each \( p = (x, T, y, T') \in E \), \( U(p) : V(T) \hookrightarrow V(T') \) is a precedence function; we require, for each \( T \in \mathcal{T} \), \( v \in V(T) \), \( U(s, T, e, T)(v) = v \)
Semantics of PPO’s

- **An event type** is a pair \((x, T)\), where \(x \in \{s, e\}\) and \(T \in T\); \(E(A)\) denotes the set of event types of PPO \(A\).

- **An event** is a pair \((A, v)\), where \(A = (x, T)\) is an event type and \(v \in V(T)\); \(ev(A)\) denotes the set of events of PPO \(A\).

- **Event** \((B, w)\) is an immediate predecessor of event \((A, v)\), notation \((B, w) \rightarrow (A, v)\), if \((B, A) \in E \land U(B, A)(w) = v\).

- Let \(C \subseteq ev(A)\) and \(\alpha \in ev(A)\) with \(\alpha \notin C\). Then \(C\) enables \(\alpha\), notation \(C \vdash \alpha\), if all immediate predecessors of \(\alpha\) are in \(C\).
The set $\text{conf}(\mathcal{A})$ of configurations of a PPO $\mathcal{A}$ is the smallest subset of $2^{\text{ev}(\mathcal{A})}$ such that:

1. $\emptyset \in \text{conf}(\mathcal{A})$,
2. if $C \in \text{conf}(\mathcal{A})$, and $C \vdash \alpha$ then $C \cup \{\alpha\} \in \text{conf}(\mathcal{A})$.
The set $\text{conf}(\mathcal{A})$ of configurations of a PPO $\mathcal{A}$ is the smallest subset of $2^{\text{ev}(\mathcal{A})}$ such that:

1. $\emptyset \in \text{conf}(\mathcal{A})$,
2. if $C \in \text{conf}(\mathcal{A})$, and $C \vdash \alpha$ then $C \cup \{\alpha\} \in \text{conf}(\mathcal{A})$

The configuration structure of $\mathcal{A}$ is the labeled transition system $\mathcal{C}(\mathcal{A}) = (\text{conf}(\mathcal{A}), \emptyset, \mathcal{E}(\mathcal{A}), \rightsquigarrow)$, where $C \overset{\mathcal{A}}{\rightsquigarrow} C \cup \{\alpha\}$ iff $C \in \text{conf}(\mathcal{A})$, $\alpha$ has event type $\mathcal{A}$, and $C \vdash \alpha$. 
Semantics of PPO’s (cnt)

The set $\text{conf}(A)$ of configurations of a PPO $A$ is the smallest subset of $2^{\text{ev}(A)}$ such that:

1. $\emptyset \in \text{conf}(A)$,
2. if $C \in \text{conf}(A)$, and $C \vdash \alpha$ then $C \cup \{\alpha\} \in \text{conf}(A)$

The configuration structure of $A$ is the labeled transition system $C(A) = (\text{conf}(A), \emptyset, \mathcal{E}(A), \rightsquigarrow)$, where $C \overset{A}{\rightsquigarrow} C \cup \{\alpha\}$ iff $C \in \text{conf}(A)$, $\alpha$ has event type $A$, and $C \vdash \alpha$

Cf work of Winskel (1989) and Van Glabbeek & Plotkin (2009) on configuration structures, event structures and Petri nets
An event is reachable if it occurs in some configuration of $\mathcal{A}$. For $\alpha, \beta$ reachable events, write $\alpha \leq_{\mathcal{A}} \beta$, if for each $C \in \text{conf}(\mathcal{A})$, $\beta \in C$ implies $\alpha \in C$.

**Lemma**

$\leq_{\mathcal{A}}$ is a partial order on the set of reachable events of a PPO $\mathcal{A}$.
Example

Fred Houben, Georgeta Igna and Frits Vaandrager

Modeling Task Systems Using Parameterized Partial Orders
Translation to Uppaal

We translate each restricted PPO $A$ to a network of Uppaal automata that contains, for each task $T$, an automaton

\[
\text{!done}[T] && \text{dep\_met}(\text{start}(T)) \\
\text{update()} \\
\text{end}(T) ? \text{start}(T)! \\
\text{L1} \\
\text{L2} \\
\text{dep\_met}(\text{end}(T))
\]

**Theorem**

Let $A$ be a restricted PPO. Then $\mathcal{C}(A)$ is isomorphic to the reachable part of the labeled transition system induced by the associated Uppaal network.
Extending the Translation with Architecture and Mapping

Fred Houben, Georgeta Igna and Frits Vaandrager

Modeling Task Systems Using Parameterized Partial Orders
Case Study + Experiments

How do analysis times for Uppaal models generated via translation compare to those for handcrafted models?

Fred Houben, Georgeta Igna and Frits Vaandrager

Modeling Task Systems Using Parameterized Partial Orders
Use Cases Process From Store and Simple Print

Fred Houben, Georgeta Igna and Frits Vaandrager
Modeling Task Systems Using Parameterized Partial Orders
<table>
<thead>
<tr>
<th>DC</th>
<th>SP</th>
<th>Mem (KB)</th>
<th>Time (s)</th>
<th>Make-span(s)</th>
<th>States Explored</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4500</td>
<td>0.50</td>
<td>23</td>
<td>1130</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5124</td>
<td>0.40</td>
<td>23</td>
<td>413</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>5480</td>
<td>1.60</td>
<td>71</td>
<td>10578</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5402</td>
<td>1.10</td>
<td>71</td>
<td>3050</td>
</tr>
<tr>
<td>35</td>
<td>50</td>
<td>12808</td>
<td>11.31</td>
<td>367</td>
<td>149206</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9184</td>
<td>13.51</td>
<td>367</td>
<td>48196</td>
</tr>
<tr>
<td>70</td>
<td>100</td>
<td>23568</td>
<td>27.92</td>
<td>737</td>
<td>433816</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18016</td>
<td>43.74</td>
<td>737</td>
<td>155491</td>
</tr>
<tr>
<td>334</td>
<td>500</td>
<td>593996</td>
<td>279.70</td>
<td>3585</td>
<td>6843592</td>
</tr>
<tr>
<td></td>
<td></td>
<td>282220</td>
<td>898.98</td>
<td>3585</td>
<td>3038099</td>
</tr>
<tr>
<td>667</td>
<td>1000</td>
<td>2321768</td>
<td>1304.87</td>
<td>7166</td>
<td>25206064</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1076196</td>
<td>3702.10</td>
<td>7166</td>
<td>11704000</td>
</tr>
<tr>
<td>903</td>
<td>1355</td>
<td>4168896</td>
<td>1937.88</td>
<td>9705</td>
<td>45226661</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1962636</td>
<td>7155.40</td>
<td>9705</td>
<td>21272017</td>
</tr>
<tr>
<td>904</td>
<td>1356</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1964952</td>
<td>7173.70</td>
<td>9715</td>
<td>21302397</td>
</tr>
<tr>
<td>1460</td>
<td>1960</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4053164</td>
<td>17055.36</td>
<td>15117</td>
<td>44117751</td>
</tr>
</tbody>
</table>

**Figure:** Handcrafted models (grey) vs. Generated models
Experiments: Direct Copy (DC) || Process from Store (PFS)

<table>
<thead>
<tr>
<th>DC</th>
<th>PFS</th>
<th>Mem (KB)</th>
<th>Time (s)</th>
<th>Make-span(s)</th>
<th>States Explored</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4456</td>
<td>0.40</td>
<td>15</td>
<td>704</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>7540</td>
<td>4.10</td>
<td>114</td>
<td>47561</td>
</tr>
<tr>
<td>25</td>
<td>80</td>
<td>21352</td>
<td>19.51</td>
<td>279</td>
<td>334606</td>
</tr>
<tr>
<td>120</td>
<td>240</td>
<td>586172</td>
<td>384.66</td>
<td>1324</td>
<td>8255421</td>
</tr>
<tr>
<td></td>
<td></td>
<td>244230</td>
<td>991.61</td>
<td>1324</td>
<td>3269408</td>
</tr>
<tr>
<td>240</td>
<td>480</td>
<td>2555392</td>
<td>1857.78</td>
<td>2844</td>
<td>33327361</td>
</tr>
<tr>
<td>303</td>
<td>606</td>
<td>1007540</td>
<td>4245.12</td>
<td>2844</td>
<td>13162088</td>
</tr>
<tr>
<td>304</td>
<td>608</td>
<td>6077452</td>
<td>2419.45</td>
<td>3337</td>
<td>43223828</td>
</tr>
<tr>
<td>1572420</td>
<td>7053.88</td>
<td>3337</td>
<td>21007415</td>
<td></td>
<td></td>
</tr>
<tr>
<td>304</td>
<td>608</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
</tr>
<tr>
<td>1582848</td>
<td>7187.23</td>
<td>3348</td>
<td>21146634</td>
<td></td>
<td></td>
</tr>
<tr>
<td>460</td>
<td>960</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
<td>O.M.</td>
</tr>
<tr>
<td>4056016</td>
<td>20827.78</td>
<td>5284</td>
<td>52819448</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: Handcrafted models (grey) vs. Generated models
Conclusions and Future Work

1. PPOs are simple but expressive extension of task graphs
2. We presented translation to Uppaal and proved correctness
3. Models generated through translation are more tractable than handcrafted models
4. State space explosions remains issue
5. Future work:
   ▶ exploit structure PPOs to alleviate state space explosions
   ▶ stability of greedy schedules