An SMT-based Approach to Fair Termination Analysis

Javier Esparza, Philipp J. Meyer

Technische Universität München
Fair Termination Analysis

- Fair termination: No non-fair infinite execution sequence $\sigma$.
- PSPACE-complete for boolean programs.
Fair Termination Analysis

- Fair termination: No non-fair infinite execution sequence $\sigma$.
- PSPACE-complete for boolean programs.

SMT-Based Approach

- Incomplete method based on reduction to feasibility of linear arithmetic constraints.
- Strengthened with refinement cycle which adds mixed linear and boolean constraints.
- Similar method previously applied for safety properties (An SMT-based Approach to Coverability Analysis, CAV14).
Lamport’s 1-bit Algorithm for Mutual Exclusion

\begin{align*}
\textbf{procedure} \text{ Process 1} & \\
\text{begin} & \\
\quad b_1 & := 0 \\
\quad \textbf{while} \ true \ \textbf{do} & \\
\quad p_1: & b_1 := 1 \\
\quad p_2: & \textbf{while} b_2 = 1 \ \textbf{do} \text{ skip} \ \textbf{od} \\
\quad p_3: & (\ast \text{ critical section } \ast) \\
\quad & b_1 := 0 \\
\quad \textbf{od} & \\
\text{end} & \\
\textbf{procedure} \text{ Process 2} & \\
\text{begin} & \\
\quad b_2 & := 0 \\
\quad \textbf{while} \ true \ \textbf{do} & \\
\quad q_1: & b_2 := 1 \\
\quad q_2: & \textbf{if} b_1 = 1 \ \textbf{then} \\
\quad q_3: & b_2 := 0 \\
\quad q_4: & \textbf{while} b_1 = 1 \ \textbf{do} \text{ skip} \ \textbf{od} \\
\quad & \textbf{goto} q_1 \\
\quad q_5: & (\ast \text{ critical section } \ast) \\
\quad & b_2 := 0 \\
\quad \textbf{od} & \\
\text{end} &
\end{align*}
Property: If both processes are executed infinitely often, then the first process should enter the critical section ($p_3$) infinitely often.
Property: If both processes are executed infinitely often, then the first process should enter the critical section ($p_3$) infinitely often.
Property: If both processes are executed infinitely often, then the first process should enter the critical section ($p_3$) infinitely often.
Abstract View of the Model

Property: For every infinite transition sequence $\sigma$, we have
\[
\varphi(\sigma) = \bigvee_{i=1}^{4} (s_i \in \inf(\sigma)) \land \bigvee_{i=1}^{7} (t_i \in \inf(\sigma)) \implies s_2 \in \inf(\sigma).
\]
Loop Sequences

\[ \{ p_1, nb_1, nb_2, q_1 \} \xrightarrow{t_1 t_6 t_7 s_1 t_1 t_2 t_3 s_2 t_5 s_3 t_4} \{ p_1, nb_1, nb_2, q_1 \} \]
Loop Sequences

\[ \{p_1, nb_1, nb_2, q_1\} \xrightarrow{t_1 t_6 t_7 s_1 t_1 t_2 t_3 s_2 t_5 s_3 t_4} \{p_1, nb_1, nb_2, q_1\} \]

\[ #\sigma = (\#t_1 \ #t_2 \ #t_3 \ #t_4 \ #t_5 \ #t_6 \ #t_7 \ #s_1 \ #s_2 \ #s_3 \ #s_4) \]
Loop Sequences

\[
\{p_1, nb_1, nb_2, q_1\} \xrightarrow{t_1 t_6 t_7 s_1 t_1 t_2 t_3 s_2 t_5 s_3 t_4} \{p_1, nb_1, nb_2, q_1\}
\]

\[
\#\sigma = \begin{pmatrix} #t_1 & #t_2 & #t_3 & #t_4 & #t_5 & #t_6 & #t_7 & #s_1 & #s_2 & #s_3 & #s_4 \end{pmatrix} = \begin{pmatrix} 2 \end{pmatrix}
\]
Loop Sequences

\[ \{ p_1, nb_1, nb_2, q_1 \} \xrightarrow{t_1 t_6 t_7 s_1 t_1 t_2 t_3 s_2 t_5 s_3 t_4} \{ p_1, nb_1, nb_2, q_1 \} \]

\[ \#\sigma = \begin{pmatrix} \#t_1 & \#t_2 & \#t_3 & \#t_4 & \#t_5 & \#t_6 & \#t_7 & \#s_1 & \#s_2 & \#s_3 & \#s_4 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \]
Loop Sequences

\[ \{p_1, nb_1, nb_2, q_1\} \xrightarrow{t_1t_6t_7s_1t_1t_2t_3s_2t_5s_3t_4} \{p_1, nb_1, nb_2, q_1\} \]

\[ \#\sigma = (\begin{array}{cccccccccccc} #t_1 & #t_2 & #t_3 & #t_4 & #t_5 & #t_6 & #t_7 & #s_1 & #s_2 & #s_3 & #s_4 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}) \]
Necessary Condition for Loops

\[ X = \begin{pmatrix} \#t_1 & \#t_2 & \#t_3 & \#t_4 & \#t_5 & \#t_6 & \#t_7 & \#s_1 & \#s_2 & \#s_3 & \#s_4 \\ t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4 \end{pmatrix} \]
Necessary Condition for Loops

\[ X = \left( \begin{array}{cccccccc}
    #t_1 & #t_2 & #t_3 & #t_4 & #t_5 & #t_6 & #t_7 & #s_1 & #s_2 & #s_3 & #s_4 \\
    t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4
  \end{array} \right) \]
Necessary Condition for Loops

\[ X = \begin{pmatrix} #t_1 & #t_2 & #t_3 & #t_4 & #t_5 & #t_6 & #t_7 & #s_1 & #s_2 & #s_3 & #s_4 \\ t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4 \end{pmatrix} \]

\[ q_1 : \quad t_4 + t_7 = t_1 \]
Necessary Condition for Loops

\[ X = \begin{pmatrix}
    \#t_1 & \#t_2 & \#t_3 & \#t_4 & \#t_5 & \#t_6 & \#t_7 & \#s_1 & \#s_2 & \#s_3 & \#s_4 \\
    t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4
\end{pmatrix} \]

\[ q_1 : \quad t_4 + t_7 = t_1 \]
Necessary Condition for Loops

\[ X = \begin{pmatrix}
#t_1 & #t_2 & #t_3 & #t_4 & #t_5 & #t_6 & #t_7 & #s_1 & #s_2 & #s_3 & #s_4 \\
 t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4
\end{pmatrix} \]

\[ q_1 : \quad t_4 + t_7 = t_1 \]

\[ q_2 : \quad t_1 = t_2 + t_6 \]

Diagram of a loop with transitions labeled with \( t_1, t_2, t_3, t_4, t_5, t_6, t_7 \) and states labeled with \( q_1, q_2, q_3, q_4, q_5 \).
Necessary Condition for Loops

\[ X = \begin{pmatrix}
#t_1 & #t_2 & #t_3 & #t_4 & #t_5 & #t_6 & #t_7 & #s_1 & #s_2 & #s_3 & #s_4 \\
t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4
\end{pmatrix} \]

\[ q_1 : \quad t_4 + t_7 = t_1 \]
\[ q_2 : \quad t_1 = t_2 + t_6 \]
\[ q_3 : \quad t_2 = t_3 \]
\[ q_4 : \quad t_3 = t_4 \]
\[ q_5 : \quad t_6 = t_7 \]
Necessary Condition for Loops

\[ p_1 : \ s_3 = s_1 \]
\[ p_2 : \ s_1 = s_2 \]
\[ p_3 : \ s_2 = s_3 \]

\[ b_2 : \ t_1 = t_3 + t_7 \]
\[ nb_2 : \ t_3 + t_7 = s_1 \]

\[ b_1 : \ s_1 = s_3 \]
\[ nb_1 : \ s_3 = s_1 \]

\[ q_1 : \ t_4 + t_7 = t_1 \]
\[ q_2 : \ t_1 = t_2 + t_6 \]
\[ q_3 : \ t_2 = t_3 \]
\[ q_4 : \ t_3 = t_4 \]
\[ q_5 : \ t_6 = t_7 \]
Termionation Constraints

- Accumulate constraints in matrix form as $C \cdot X = 0$.
- If there is an infinite transition sequence $\sigma$, then the following constraints have a solution $X$:

\[
\mathcal{C} :: \begin{cases} 
C \cdot X = 0 \\
X \geq 0 \\
X \neq 0
\end{cases}
\]

- If the constraints have no solution, then the program is terminating.
- A solution $X$ is realizable if there is a sequence $\sigma$ with $\#\sigma = X$. 
Fair Termination Constraints

- Fairness condition given by boolean formula $\varphi$ over $t \in \inf(\sigma)$.
- If the program is not fairly terminating, then there is an infinite transition sequence $\sigma$ satisfying $\sigma \models \neg \varphi$.
- Add constraint $\neg \varphi(X)$ to $\mathcal{C}$ for fair termination constraints.

Fairness for Lamport’s Algorithm

$$\varphi(\sigma) = \bigvee_{i=1}^{4} (s_i \in \inf(\sigma)) \land \bigvee_{i=1}^{7} (t_i \in \inf(\sigma)) \implies s_2 \in \inf(\sigma)$$

$$\neg \varphi(X) = (s_1 + s_2 + s_3 + s_4 > 0) \land (t_1 + t_3 + t_4 + t_5 + t_6 + t_7 > 0) \land (s_2 = 0)$$
Fair Termination Constraints

\[ s_3 = s_1 \quad t_4 + t_7 = t_1 \quad s_1 \geq 0 \quad t_1 \geq 0 \]
\[ s_1 = s_2 \quad t_1 = t_2 + t_6 \quad s_2 \geq 0 \quad t_2 \geq 0 \]
\[ s_2 = s_3 \quad t_2 = t_3 \quad s_3 \geq 0 \quad t_3 \geq 0 \]
\[ t_3 = t_4 \quad t_4 \geq 0 \]
\[ t_6 = t_7 \quad t_5 \geq 0 \]
\[ s_1 = s_3 \quad t_1 = t_3 + t_7 \quad t_6 \geq 0 \]
\[ s_3 = s_1 \quad t_3 + t_7 = s_1 \quad t_7 \geq 0 \]

\[ s_1 + s_2 + s_3 + s_4 + t_1 + t_3 + t_4 + t_5 + t_6 + t_7 > 0 \]

\[ (s_1 + s_2 + s_3 + s_4 > 0) \land \]
\[ (t_1 + t_3 + t_4 + t_5 + t_6 + t_7 > 0) \land \]
\[ (s_2 = 0) \]
Fair Termination Constraints: Solution

\[ X = \begin{pmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ s_3 = s_1 \quad t_4 + t_7 = t_1 \quad s_1 \geq 0 \quad t_1 \geq 0 \]
\[ s_1 = s_2 \quad t_1 = t_2 + t_6 \quad s_2 \geq 0 \quad t_2 \geq 0 \]
\[ s_2 = s_3 \quad t_2 = t_3 \quad s_3 \geq 0 \quad t_3 \geq 0 \]
\[ t_3 = t_4 \quad t_6 = t_7 \quad t_4 \geq 0 \quad t_5 \geq 0 \]
\[ s_1 = s_3 \quad t_1 = t_3 + t_7 \quad t_6 \geq 0 \]
\[ s_3 = s_1 \quad t_3 + t_7 = s_1 \quad t_7 \geq 0 \]

\[ s_1 + s_2 + s_3 + s_4 + t_1 + t_3 + t_4 + t_5 + t_6 + t_7 > 0 \]

\[ (s_1 + s_2 + s_3 + s_4 > 0) \land \]
\[ (t_1 + t_3 + t_4 + t_5 + t_6 + t_7 > 0) \land \]
\[ (s_2 = 0) \]
Fair Termination Constraints: Solution

\[ X = \begin{pmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & s_1 & s_2 & s_3 & s_4 \end{pmatrix} \]

\( X = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \)
Solution realizable?

\( X \) realized by \( \sigma \) with \( \inf(\sigma) = \{s_4, t_5\} \).
Solution realizable?

$X$ realized by $\sigma$ with $\text{inf}(\sigma) = \{s_4, t_5\}$. 

---

Diagram: 

- $p_1$ to $s_1$ 
- $p_2$ to $s_4$ 
- $p_3$ to $s_2$ 
- $b_1$ to $s_1$ and back to $b_1$ 
- $nb_1$ to $s_3$ and back to $nb_1$ 
- $t_4$, $t_6$ 
- $q_1$ to $t_7$ 
- $q_2$ to $t_6$ 
- $q_3$ to $t_3$ 
- $q_4$ to $t_5$ 
- $t_2$, $t_5$ 
- $t_4$, $t_6$ 
- $s_4$ to $t_1$ and $t_3$, $t_7$ 
- $nb_2$ to $s_2$ and back to $nb_2$ 
- $s_2$ to $t_1$ and $t_3$, $t_7$ 
- $s_3$ to $p_1$ and $p_2$ 
- $s_4$ to $p_2$ and $p_3$ 
- $s_2$ to $p_3$ 
- $t_2$, $t_5$
Refinement Component

$q_1$, $q_4$ and $b_2$ are in mutual exclusion.
Refinement Component

$q_1$, $q_4$ and $b_2$ are in mutual exclusion.
Refinement Component

$q_1$, $q_4$ and $b_2$ are in mutual exclusion.
Refinement Component

$q_1, q_4$ and $b_2$ are in mutual exclusion.
Refinement Constraint

\( X \) realized by \( \sigma \) with \( \inf(\sigma) = \{s_4, t_5\} \).
Refinement Constraint

$X$ not realizable $\Rightarrow$ Generate refinement constraint $\delta$. 
Refinement Constraint

$$\delta = (s_4 = 0) \lor (t_5 = 0) \lor (t_1 + t_3 + t_4 + t_7 > 0)$$
Refinement Loop

$\mathcal{C}$ sat?

Obtain solution $X$.

Refinement component to discard $X$?

Generate refinement constraint $\delta$. unsat
Refinement Loop

\[ C \text{ sat?} \rightarrow \text{unsat} \rightarrow \text{terminating} \]
Refinement Loop

\[ C \text{ sat?} \quad \overset{\text{unsat}}{\longrightarrow} \quad \text{terminating} \]

\[ \text{sat} \quad \overset{}{\longrightarrow} \quad \text{Obtain solution } X. \]
Refinement Loop

Refinement component to discard $X$?

Obtain solution $X$.

$\mathcal{C}$ sat? \(\xrightarrow{\text{unsat}}\) terminating \(\xrightarrow{\text{sat}}\)

Refinement Loop

\( C \) sat? → unsat → terminating

\( C \) sat? → sat

Obtain solution \( X \).

inconclusive ← no

Refinement component to discard \( X \)?
Refinement Loop

\[ C \text{ sat?} \]

- \text{unsat} \rightarrow \text{terminating}
- \text{sat}

Obtain solution \( X \).

Generate refinement constraint \( \delta \).

\text{yes} \rightarrow \text{inconclusive}

\text{no} \rightarrow \text{Refinement component to discard } X?
Refinement Loop

\[ \mathcal{C} := \mathcal{C} \cup \{ \delta \} \]

Generate refinement constraint \( \delta \).

\( \mathcal{C} \) sat?

Unsat \( \rightarrow \) terminating

Sat

Obtain solution \( X \).

Refinement component to discard \( X \)?

Yes \( \rightarrow \) inconclusive

No \( \rightarrow \) yes

Yes

inconclusive

no

Refinement component to discard \( X \)?
Experimental Evaluation

Benchmarks

- IBM/SAP — Workflow nets from business process models
  - 1976 examples
  - 1836 terminating

- Erlang — Models from the verification of Erlang programs
  - 50 examples, up to 66950 places and 213626 transitions
  - 33 terminating

- Literature — Selected examples from the literature
  - 5 examples, with unbounded variables
  - All terminating

- Classical — Classic asynchronous programs for mutual exclusion and distributed algorithms
  - 5 examples, scalable in number of processes
  - All fairly terminating
Rate of Success

<table>
<thead>
<tr>
<th></th>
<th>1264</th>
<th>572</th>
<th>33</th>
<th>5</th>
<th>5</th>
<th>1879</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erlang</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Literature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rate of Success

- IBM: 1264 terminating, 1263 w/o refinement
- SAP: 572 terminating, 571 w/o refinement
- Erlang: 33 terminating, 27 w/o refinement
- Literature: 5 terminating, 0 w/o refinement
- Classical: 5 terminating, 0 w/o refinement
- Total: 1879 terminating, 1861 w/o refinement
Rate of Success

<table>
<thead>
<tr>
<th></th>
<th>IBM</th>
<th>SAP</th>
<th>Erlang</th>
<th>Literature</th>
<th>Classical</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1264</td>
<td>1264</td>
<td>572</td>
<td>33</td>
<td>5</td>
<td>5</td>
<td>1879</td>
</tr>
<tr>
<td>1263</td>
<td>1263</td>
<td>571</td>
<td>27</td>
<td>5</td>
<td>5</td>
<td>1873</td>
</tr>
<tr>
<td>1861</td>
<td></td>
<td></td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>1861</td>
</tr>
</tbody>
</table>

IBM: terminating w/o refinement w/o refinement
SAP: terminating w/o refinement with refinement
Erlang: terminating w/o refinement with refinement
Literature: terminating with refinement
Classical: terminating with refinement
Total: terminating w/o refinement with refinement
Performance on Positive Examples

Execution time (s) vs Number of places graph with a time limit marker.
Performance on Positive Examples

![Graph showing execution time vs. number of places. The x-axis represents the number of places on a logarithmic scale, ranging from $10^0$ to $10^5$. The y-axis represents execution time (s) on a logarithmic scale, ranging from $10^{-2}$ to $10^4$. There are data points indicating execution times of 3 seconds, 11 seconds, and 8 minutes, which are marked with red circles. The graph also includes a red horizontal line representing the time limit.](image-url)
Performance on Negative Examples

![Graph showing execution time vs. number of places]
Performance on Negative Examples

![Graph showing the relationship between execution time (s) and number of places with a time limit of 5 seconds and 1000 places as constraints.](image)
Refinement Steps

![Graph showing refinement steps vs. number of places.](image_url)
Comparison with SPIN on Scaled Classical Suite

![Graph comparing SPIN and Petrinizer performance on various classical algorithms.](image)

- SPIN (s)
- Petrinizer (s)

- Leader Election
- Snapshot
- Lamport
- Peterson
- Szymanski

- time limit/out of memory
Summary

- Fast and effective technique for proving fair termination
- Incomplete, but high degree of completeness
- Large instances can be handled
- Constraints can be used as a certificate of fair termination