Alcino Cunha

SPECIFICATION AND MODELING

BOUNDED MODEL CHECKING

Universidade do Minho & INESC TEC

2019/20

INFINITE TRACES

WHY INFINITE TRACES ONLY?

Semantic issues with finite traces

- What formulas are true in the last state?
- Different possible semantics

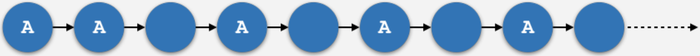
In Electrum: infinite traces only

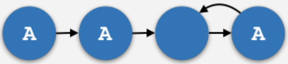
- Caveat: no deadlock detection in general
- But a finite trace can be represented by an infinite one stuttering on the last state

FINITE REPRESENTATION WITH LASSOS

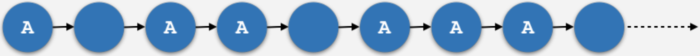
Lasso trace

- Some infinite traces can be represented by finite *lasso traces*





- Notice some infinite traces cannot



Small-Model Property for LTL

If an LTL formula is satisfiable, then it is satisfied by at least one lasso trace.

BOUNDED MODEL CHECKING

BOUNDED MODEL CHECKING

- Given a model of a system with
  - I, a state formula with no primes and no temporal operators
  - T, a transition formula with no temporal operators

```
fact { I and always T }
```

- A BMC procedure verifies that a LTL formula ϕ is valid for all lasso traces of such model of size up to k

```
check { φ } for ... but k steps
```

REDUCING VALIDITY TO (UN)SATISFIABILITY

- To verify a formula ϕ it suffices to search for a counter-example
  - A lasso trace satisfying **not** ϕ
  - If no counter-example exists the formula is valid

run { **not** ϕ } **for** ... **but** k **steps**

- All sizes from 1 to k are searched iteratively
  - So that counter-examples of minimal size are returned

run { **not** ϕ } **for** ... **but exactly** 1 **steps**
run { **not** ϕ } **for** ... **but exactly** 2 **steps**
...
run { **not** ϕ } **for** ... **but exactly** k **steps**

ENCODING LASSO TRACES OF SIZE k IN KODKOD

- Every mutable relation r originates k (static) relations r_0 to r_{k-1}
  - The vector of all relations at state i will be denoted by $\overline{r_i}$
  - Initial state formula I is defined over $\overline{r}$
  - The transition formula T is defined over $\overline{r}$ and $\overline{r'}$
- A lasso trace of model I, T with reentry at state $0 \leq l < k$ is specified by formula

$$_l[\![I, T]\!]_k \quad \equiv \quad I[\overline{r} \leftarrow \overline{r_0}] \quad \wedge \bigwedge_{0 \leq i < k-1} T[\overline{r} \leftarrow \overline{r_i}, \overline{r'} \leftarrow \overline{r_{i+1}}] \quad \wedge \quad T[\overline{r} \leftarrow \overline{r_{k-1}}, \overline{r'} \leftarrow \overline{r_l}]$$

- The following formula can be used to generate an arbitrary lasso trace of size k of the model I, T

$$\bigvee_{l=0}^{k-1} {}_l[\![I, T]\!]_k$$

TRASH EXAMPLE

```
var sig File {}
var sig Trash in File {}

fact {
  no Trash
  always (
    -- empty trash or ...
    (some Trash and no Trash' and File' = File - Trash) or ...
  )
}

run {} for 3 but exactly 3 steps
```

KODKOD TRANSLATION

```
File0  :1 {} {(A),(B),(C)}
File1  :1 {} {(A),(B),(C)}
File2  :1 {} {(A),(B),(C)}
Trash0 :1 {} {(A),(B),(C)}
Trash1 :1 {} {(A),(B),(C)}
Trash2 :1 {} {(A),(B),(C)}

-- sub-typing constraints
-- always Trash in File
Trash0 in File0 and Trash1 in File1 and Trash2 in File2
```

KODKOD TRANSLATION

```
-- initial state
no Trash0

-- transitions
(some Trash0 and no Trash1 and File1 = File0 - Trash0) or ...
(some Trash1 and no Trash2 and File2 = File1 - Trash1) or ...

-- loop
((some Trash2 and no Trash0 and File0 = File2 - Trash2) or ...) or
((some Trash2 and no Trash1 and File1 = File2 - Trash2) or ...) or
((some Trash2 and no Trash2 and File2 = File2 - Trash2) or ...)
```

LINEAR TEMPORAL RELATIONAL LOGIC ABSTRACT SYNTAX

$$
\begin{array}{rcll}
\phi & ::= & \Phi_1 \subseteq \Phi_2 & \\
 & | & \neg\phi & \\
 & | & \phi_1 \wedge \phi_2 & \\
 & | & \mathsf{G}\,\phi & \\
 & | & \mathsf{F}\,\phi & \\
 & | & \mathsf{X}\,\phi & \\
 & | & \ldots & \\
 & & & \\
\Phi & ::= & R & \\
 & | & \Phi_1 \cup \Phi_2 & \mathrm{ar}(\Phi) = \mathrm{ar}(\Psi) \\
 & | & \Phi_1 \,.\, \Phi_2 & \mathrm{ar}(\Phi) + \mathrm{ar}(\Psi) > 2 \\
 & | & \Phi' & \\
 & | & \ldots &
\end{array}
$$

ENCODING TEMPORAL FORMULAS IN LASSO TRACES

- Unroll $\mathsf{G}\,\phi$ and $\mathsf{F}\,\phi$ to check ϕ in different state of the trace
  - The states to be checked depend on the current state and the re-entry state
- For $\mathsf{X}\,\phi$ and Φ', evaluate ϕ and Φ in the successor state

$$succ(i) = \begin{cases} i+1 & \text{if } i < k-1 \\ l & \text{if } i = k-1 \end{cases}$$

ENCODING TEMPORAL FORMULAS IN LASSO TRACES

$$
\begin{aligned}
{}_l[\![\phi]\!]_k &\equiv {}_l[\![\phi]\!]_k^0 \\[1em]
{}_l[\![\Phi \subseteq \Psi]\!]_k^i &\equiv [\![\Phi]\!]^i \subseteq [\![\Psi]\!]^i \\
{}_l[\![\neg\phi]\!]_k^i &\equiv \neg\, {}_l[\![\phi]\!]_k^i \\
{}_l[\![\phi \wedge \psi]\!]_k^i &\equiv {}_l[\![\phi]\!]_k^i \wedge {}_l[\![\psi]\!]_k^i \\
{}_l[\![\mathsf{G}\,\phi]\!]_k^i &\equiv \bigwedge_{j=min(i,l)}^{k-1} {}_l[\![\phi]\!]_k^j \\
{}_l[\![\mathsf{F}\,\phi]\!]_k^i &\equiv \bigvee_{j=min(i,l)}^{k-1} {}_l[\![\phi]\!]_k^j \\
{}_l[\![\mathsf{X}\,\phi]\!]_k^i &\equiv {}_l[\![\phi]\!]_k^{succ(i)} \\[1em]
[\![R]\!]^i &\equiv R_i \\
[\![\Phi \cup \Psi]\!]^i &\equiv [\![\Phi]\!]^i \cup [\![\Psi]\!]^i \\
[\![\Phi \,.\, \Psi]\!]^i &\equiv [\![\Phi]\!]^i \,.\, [\![\Psi]\!]^i \\
[\![\Phi']\!]^i &\equiv [\![\Phi]\!]^{succ(i)}
\end{aligned}
$$

PUTTING EVERYTHING TOGETHER

- A (temporal) formula ϕ is satisfiable in a lasso trace of size k in model I, T iff the following (non temporal) formula is satisfiable

$$\bigvee_{l=0}^{k-1} \left({}_{l}[\![I, T]\!]_k \wedge {}_{l}[\![\phi]\!]_k \right)$$

TRASH EXAMPLE

```
var sig File {}
var sig Trash in File {}

fact {
  no Trash
  always (
    -- empty trash or ...
    (some Trash and no Trash' and File' = File - Trash) or ...
  )
}

check {always some File} for 3 but exactly 3 steps
```

KODKOD TRANSLATION

```
-- initial state
no Trash0

-- transitions
(some Trash0 and no Trash1 and File1 = File0 - Trash0) or ...
(some Trash1 and no Trash2 and File2 = File1 - Trash1) or ...

-- loop
((some Trash2 and no Trash0 and File0 = File2 - Trash2) or ...) or
((some Trash2 and no Trash1 and File1 = File2 - Trash2) or ...) or
((some Trash2 and no Trash2 and File2 = File2 - Trash2) or ...)

-- eventually no File
no File0 or no File1 or no File2
```