Communicating Finite Automata

Communicating Finite Automata (CFA)
- concrete and abstract syntax,
- networks of CFA,
- operational semantics.

Transition Sequences
- Deadlock, Reachability
- Uppaal
  - tool demo (simulator),
  - query language,
  - CFA model-checking.

CFA at Work
- drive to configuration, scenarios, invariants
- tool demo (verifier).

Uppaal Architecture
Communicating Finite Automata presentation follows (Olderog and Dierks, 2008)

Example

```
half_idle
request_sent
tea_selected
soft_selected
water_selected

idle
DOK?
OK!

water_enabled := false, soft_enabled := false, tea_enabled := false
DTEA!
DWATER!
DSOFT!

tea_enabled
TEA?
soft_enabled
SOFT?
water_enabled
WATER?
```

ChoicePanel:
(simplified)
That is, modifications are executed sequentially from left to right.

\[ \nu \] := \nu \cdot \nu \cdot \nu \]

\( V \) = 0

By \( \Phi \)

\( \ell, \alpha, \phi, \vec{r}, \ell' \)

For each channel

\( \{ L, B, V, E, \ell \} \)

\( A \), \ldots , \( C \)

\( \tau \) /

\( \Psi \)

\( \{ \} \cup \{ \} \cup \{ \} \cup \{ \} \cup \{ \} \)

Abstract syntax

of the variables,

is a

of the variables,

is a

network

The

effect of modification

Operational Semantics of Networks of CFA
\[ \text{ℓ, ν} \text{ is reachable, i.e.} \langle \text{ℓ, ν} \rangle \in \mathcal{L} \text{ if and only if} \exists \text{ configuration } \mathcal{C}(\text{sat guards } \phi|\nu) \]
The Uppaal Query Language

Consider $N = C(A_1, \ldots, A_n)$ over data variables $V$.

- **basic formula**: $atom ::= A_i.\ell | \phi | deadlock$ where $\ell \in L_i$ is a location and $\phi$ an expression over $V$.

- **configuration formulae**: $term ::= atom | not term | term_1 and term_2$

- **existential path formulae**: $e$-formula ::= $\exists$♦ term (exists finally) $| \exists$□ term (exists globally)

- **universal path formulae**: $a$-formula ::= $\forall$♦ term (always finally) $| \forall$□ term (always globally) $| term_1 \rightarrow term_2$ (leads to)

- **formulae** (or queries): $F ::= e$-formula $| a$-formula

**Satisfaction of Uppaal Queries by Configurations**

- The satisfaction relation $\langle \vec{\ell}, \nu \rangle | = F$ between configurations $\langle \vec{\ell}, \nu \rangle = \langle (\ell_1, \ldots, \ell_n), \nu \rangle$ of a network $C(A_1, \ldots, A_n)$ and formulae $F$ of the Uppaal logic is defined inductively as follows:

  - $\langle \vec{\ell}, \nu \rangle | = deadlock$ iff $\ell_0,i$ is a deadlock configuration
  - $\langle \vec{\ell}, \nu \rangle | = A_i.\ell$ iff $\ell_0,i = \ell$
  - $\langle \vec{\ell}, \nu \rangle | = \phi$ iff $\nu | = \phi$
  - $\langle \vec{\ell}, \nu \rangle | = not term$ iff $\langle \vec{\ell}, \nu \rangle \neq term$
  - $\langle \vec{\ell}, \nu \rangle | = term_1 and term_2$ iff $\langle \vec{\ell}, \nu \rangle | = term_i$, $i = 1, 2$
Proposition. \( \neg \phi \) is reachable. Let \( \vec{\ell}, N \in F_i \) of the form \( \langle \vec{\ell}, \nu \rangle \). Example: \( \exists \xi \in F_i \) such that \( \vec{\ell} \in \xi \) holds. Satisfaction of Uppaal Queries by Configurations

\begin{align*}
\text{Satisfaction of Uppaal Queries by Configurations}
\end{align*}
Design Verification: Invariants

• Question: Is it the case that the "tea" button is only enabled if there is e1.50 in the machine? (Otherwise, the design is broken.)

• Approach: Check whether the implication tea_enabled =⇒ CoinValidator.have_c150 holds in all reachable configurations, i.e. check whether

NVM |= \forall □ (tea_enabled imply CoinValidator.have_c150)

for the vending machine model NVM.

Design Verification: Sanity Check

• Question: Is the "tea" button ever enabled? (Otherwise, the considered invariant tea_enabled =⇒ CoinValidator.have_c150 holds vacuously.)

• Approach: Check whether a configuration satisfying water_enabled = 1 is reachable. Exactly like we did with w_enabled = 0 earlier (i.e. check whether

NVM |= ∃♦ water_enabled = 1)

for the modified vending machine model NVM.'
Design Verification: Another Invariant

- drink_ready
- have_c150
- have_e1
- have_c100
- have_c50
- idle

OK?
OK?
OK?
OK?

E1?

tea_enabled := (t > 0)

C50?

water_enabled := (w > 0), tea_enabled := (t > 0)

C50?

tea_enabled := (t > 0)

C50?

soft_enabled := (s > 0)

Recall: Universal LSC Example

What Can We Conclude From Verification Results?

• Assume that query Q corresponds to a requirement on the system under development, and N is our design-idea model.
• Assume that the verification tool states N | = Q. What can we conclude from that?

N ̸| = Q
N | = Q

the design idea sat.

Q does not sat.

Q true negative

Q true positive

What can we conclude from verification results?

