Learning System Abstractions for Human Operators

Sébastien Combéfis[1] Dimitra Giannakopoulou[2]
Charles Pecheur[1] Michael Feary[2]

[1]University of Louvain (UCLouvain)
ICT, Electronics and Applied Mathematics Institute (ICTEAM)
[2]NASA Ames Research Center (ARC)

November 12, 2011

[MALETS 2011, Lawrence, KA, USA]

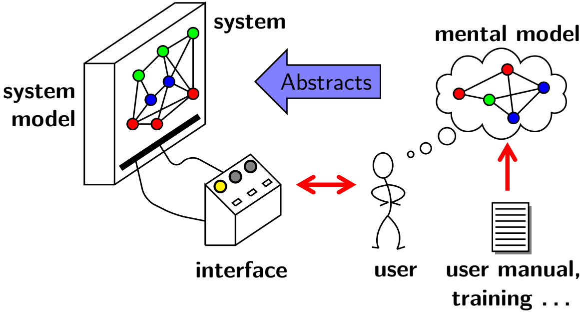
CAVEAT

This is NOT Learning

CAVEAT

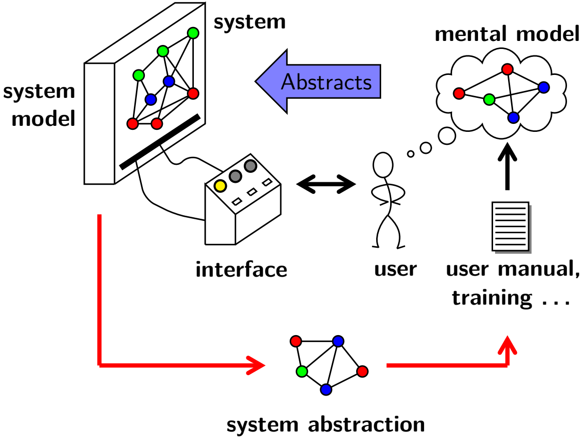
This is NOT (your usual kind of)
Learning (either)

Human-Machine Interaction



- What is a good system abstraction?

Human-Machine Interaction

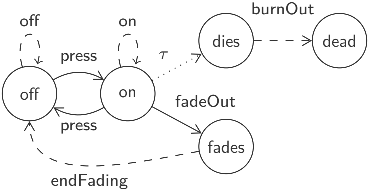


- How can such an abstraction be automatically generated?

Outline

Modelling

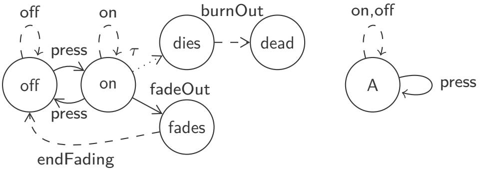
- System modelled as an HMI-LTS
- (Finite) LTS
- Commands, observations and τ



- Full-control $=$ good abstraction
- During interaction:
  - same set of commands
  - user expects all possible observations

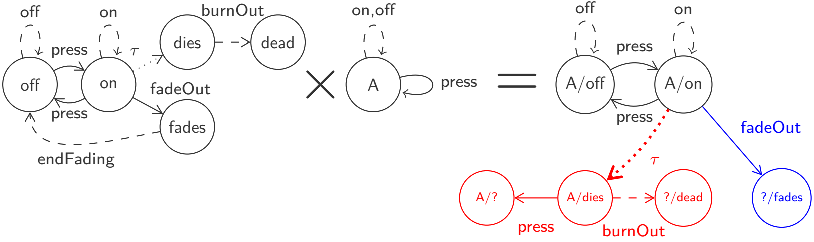
Interaction Analysis

- Interaction between a user and a system through two models:
  - System model models behaviour of the system
  - Mental model is an abstraction of the system model capturing the knowledge of the operator (conceptual model)
- The interaction is captured by the parallel execution of the two models



Interaction Analysis

- Interaction between a user and a system through two models:
  - System model models behaviour of the system
  - Mental model is an abstraction of the system model capturing the knowledge of the operator (conceptual model)
- The interaction is captured by the parallel execution of the two models



Full-control property

- Full-control property captures good system abstraction
- During the interaction between user and system:
  - The user should know exactly the available commands ...
  - ... and at least all the possible observations
- Given a system $\mathcal{M}_M = \langle S_M, s_{0_M}, \mathcal{L}^c, \mathcal{L}^o, \rightarrow_M \rangle$ and an abstraction for it $\mathcal{M}_U = \langle S_U, s_{0_U}, \mathcal{L}^c, \mathcal{L}^o, \rightarrow_U \rangle$:

$$\mathcal{M}_U \,\mathrm{fc}\, \mathcal{M}_M \textit{ iff } :$$

$$\forall \sigma \in \mathcal{L}^{co*} \text{ such that } s_{0_M} \stackrel{\sigma}{\Longrightarrow} s_M \text{ and } s_{0_U} \xrightarrow{\sigma} s_U :$$

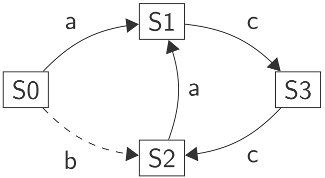
$$A^c(s_M) = A^c(s_U) \quad \wedge \quad A^o(s_M) \subseteq A^o(s_U)$$

Generation Problem

- **Goal:** Given the model of a system, automatically generate a minimal full-control system abstraction
- **Motivation:**
  - Extract the minimal behaviour of the system, so that it can be controlled without surprise
  - Help to build artifacts: manuals, procedures, trainings, . . .
  - If such abstraction does not exist, provide feedback to help redesigning the system
- Two developed algorithms : reduction-based (similarity relation) and learning-based (L^* and $3DFA$)

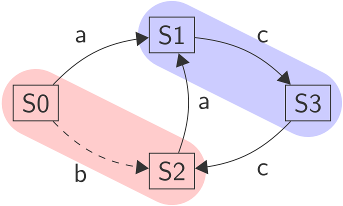
Reduction-Based Approach

- Method based on a variant of the Paige-Tarjan reduction algorithm which will partition the system by separating states which exhibit different behaviour, based on a similarity relation



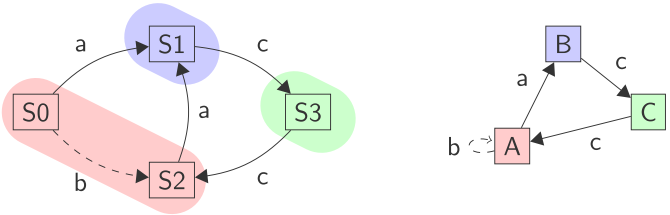
Reduction-Based Approach

- Method based on a variant of the Paige-Tarjan reduction algorithm which will partition the system by separating states which exhibit different behaviour, based on a similarity relation



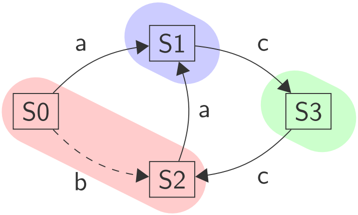
Reduction-Based Approach

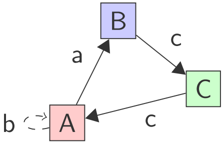
- Method based on a variant of the Paige-Tarjan reduction algorithm which will partition the system by separating states which exhibit different behaviour, based on a similarity relation



Reduction-Based Approach

- Method based on a variant of the Paige-Tarjan reduction algorithm which will partition the system by separating states which exhibit different behaviour, based on a similarity relation





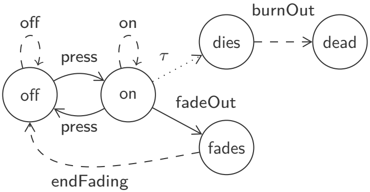
- Only works when the similarity relation is an equivalence
- Does not provide error trace when system is not proper

3-Valued Deterministic Finite Automaton

- A 3DFA is a tuple $\langle \Sigma, S, s_0, \delta, Acc, Rej, Dont \rangle$
- $\mathcal{C}^+$ denotes the DFA $\langle \Sigma, S, s_0, \delta, Acc \cup Dont \rangle$
- $\mathcal{C}^-$ denotes the DFA $\langle \Sigma, S, s_0, \delta, Acc \rangle$
- A consistent DFA $\mathcal{A}$ is such as $\mathcal{L}(\mathcal{C}^-) \subseteq \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}^+)$

Categorizing behaviour

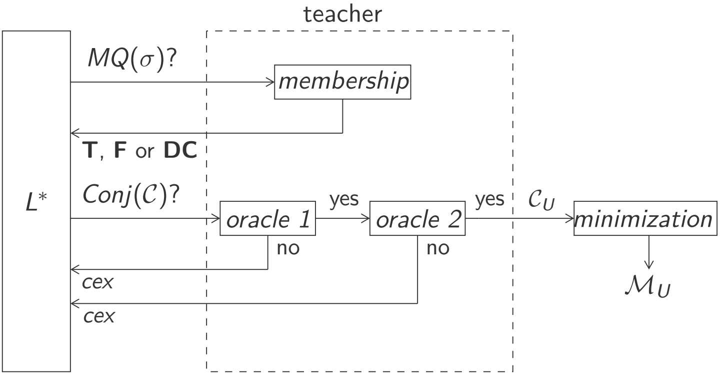
- Behaviour from the system can be categorized into three sets:
  - Accepted behaviour must be known
  - Rejected behaviour must be avoided
  - Don't care behaviour



- $\langle$ press, press $\rangle \in Acc$
- $\langle$ press, fadeOut, press $\rangle \in Rej$
- $\langle$ press, endFading $\rangle \in Dont$

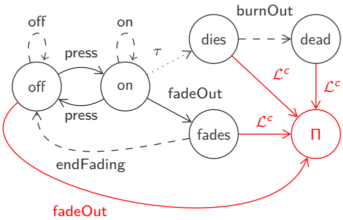
Learning-Based Approach

- Using a learning algorithm to learn a 3DFA capturing all the possible full-control system abstractions (variant of L^*)



Membership query

- **Completed system**: Adding $s \xrightarrow{\alpha} \Pi$ for each $\alpha \in \mathcal{L}^c \setminus A^c(s)$
- Given a sequence σ, it is simulated on the completed system and:
  - σ may lead to the error state: $MQ(\sigma) = \mathbf{F}$
  - σ can be simulated entirely and never leads to an error state: $MQ(\sigma) = \mathbf{T}$
  - σ cannot be simulated entirely: $MQ(\sigma) = \mathbf{DC}$



Conjecture

- Two oracles :

1. **No invalid traces**: complete $\mathcal{M}_{weak}$ on commands

$$\mathcal{C}^{+} \,||(\mathcal{M}_{weak} + s \xrightarrow{\mathcal{L}^c \backslash A^c(s)} \Pi) \quad \stackrel{\sigma}{\Longrightarrow}? \quad \Pi$$

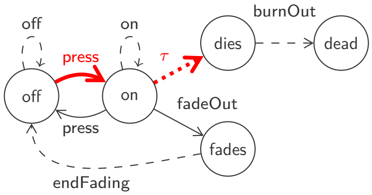
2. **All valid traces**: complete $\mathcal{C}^{-}$ on commands and observations

$$(\mathcal{C}^{-} + s \xrightarrow{\mathcal{L} \backslash A(s)} \Pi) \,||\, \mathcal{M}_{weak} \quad \stackrel{\sigma}{\Longrightarrow}? \quad \Pi$$

Full-control determinism

- System abstraction generation will fail for systems which are not full-control deterministic

- After the execution of the same trace, the enabled commands are not the same



- After executing $\langle$press$\rangle$, reaching:
  - "on" where press and fadeOut are enabled
  - "dies" where no commands are enabled

Tool

- Framework implemented within JavaPathfinder model checker
- Details presented at the JPF Workshop

Experiments

	System		Abstraction		Reduc.	Learning	
	States	Trans.	States	Trans.		3DFA states	Total
VTS	8	20	5	14	10 ms	10	92 ms
AirConditionner	154	885	27	150	177 ms	51	6 271 ms
TimedVCR	3 352	15 082	2	9	1 031 ms	6	614 ms
SimpleVCR	20	110	2	9	65 ms	6	250 ms
FullVCR	24	261	4	24	45 ms	11	432 ms
AlarmClock	42	215	5	14	–	14	512 ms
AlarmClock2	1 734	67 535	5	15	–	14	30 831 ms

- Reduction-based vs. learning-based: no clear winner
- Learning can handle more system models

Conclusion and further work

- **Conclusion**
  - A new method based on learning for generation full-control system abstraction
  - Implemented in a framework based on JavaPathfinder model checker
  - The framework can be used to detect mode confusion
- **Further work**
  - Experiment with more realistic examples
  - Experiment with variant of full-control property
    - allow the user to ignore some commands
    - integrate a "task model"
  - Revisit the reduction-based approach