A Formal Framework for Design and Analysis of Human-Machine Interaction

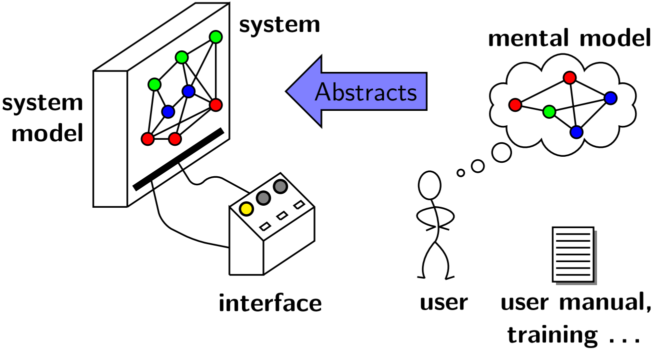
Sébastien Combéfis[1] Dimitra Giannakopoulou[2]
Charles Pecheur[1] Michael Feary[2]

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

October 11, 2011

[SMC 2011, Anchorage, AK, USA]

Human-Machine Interaction

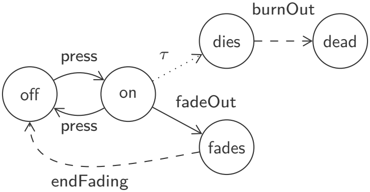


- What is a good system abstraction?
- How to automatically generate such abstractions?
- How to evaluate whether a system is well designed?

Outline

Modelling

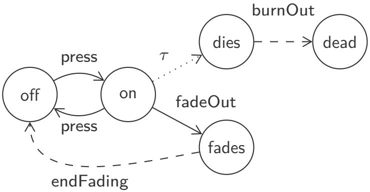
- System modelled as an HMI-LTS
- Abstracted as conceptual model
- Commands and observations

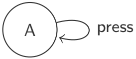


- Full-control = good abstraction
- During interaction:
  - same set of commands
  - user expects all possible observations

Interaction Analysis

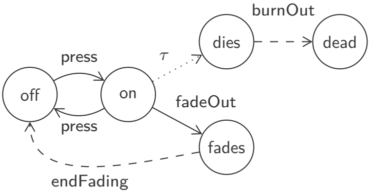
- Interaction between a user and a system through two models:
  - System model models behaviour of the system
  - Mental model is an abstraction of the system model capturing the knowledge of the operator (conceptual model)
- The interaction is captured by the parallel execution of the two models

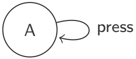


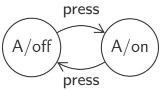


Interaction Analysis

- 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 $\mathcal{M}_U = \langle S_U, s_{0_U}, \mathcal{L}^c, \mathcal{L}^o, \rightarrow_U \rangle$:

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

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

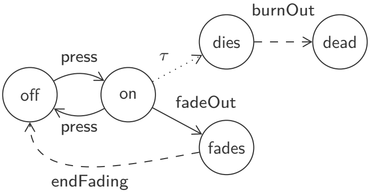
$$A^c(s_M) = A^c(s_U) \quad \wedge \quad A^o(s_M) \subseteq A^o(s_U)$$

Generation Problem

- **Goal:** Given the model of a system, automatically generate a minimal full-control 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
- Reduction-based and learning-based algorithms

Categorizing behaviour

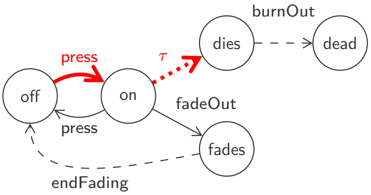
- Behaviour from the system can be categorized into three sets:
  - Accepted behaviour must be known
  - Rejected behaviour must be avoided
  - Don't care behaviour



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

Full-control determinism

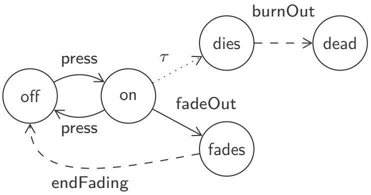
- Mental model generation will fail for systems which are not full-control deterministics
- After the execution of the same trace, the enabled commands are not the same

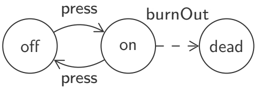


- After executing ⟨press⟩, reaching:
  - "on" where press and fadeOut are enabled
  - "dies" where no commands are enabled

Checking system against tasks

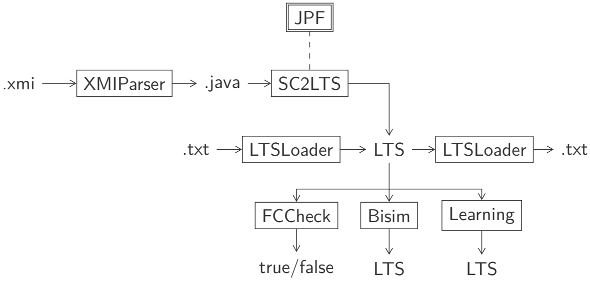
- Check whether a system covers the user tasks
- Using the full-control criterion but reversing the role of commands and observations





Framework

- Both algorithms have been implemented within Java Pathfinder (JPF) model-checker
- Systems encoded with the JPF-statechart extension
- Possibility to get models from ADEPT



Evaluation

- The methodology has been tested on two examples:
  - **Therac-25** (110 states and 312 transitions)

    Shows how mode confusion can be analyzed with our framework by adding command loops with modes
  - **Video Cassette Recorder** (1088 states and 3740 transitions)

    Shows how non-full-control-determinism can occur and how to redesign the system to solve it

Conclusion and further work

- **Conclusion**
  - Full-control property captures good abstraction
  - Methodology proposed to analyse interaction
  - Framework developed within Java Pathfinder and integration with ADEPT toolset
- **Further work**
  - Experiment with more realistic examples
  - Experiment with variant of full-control property
  - Integrate other kind of properties to be checked

Credits

- NASA, September 12, 1993, https://commons.wikimedia.org/wiki/File:1993_s51_Liftoff.jpg.