Property Preservation for Extension Patterns of State Transition Diagrams

Refinement Revisited

Christian Prehofer, fortiss GmbH

May 2015

Motivation

- State Transition diagrams to specify interactive systems

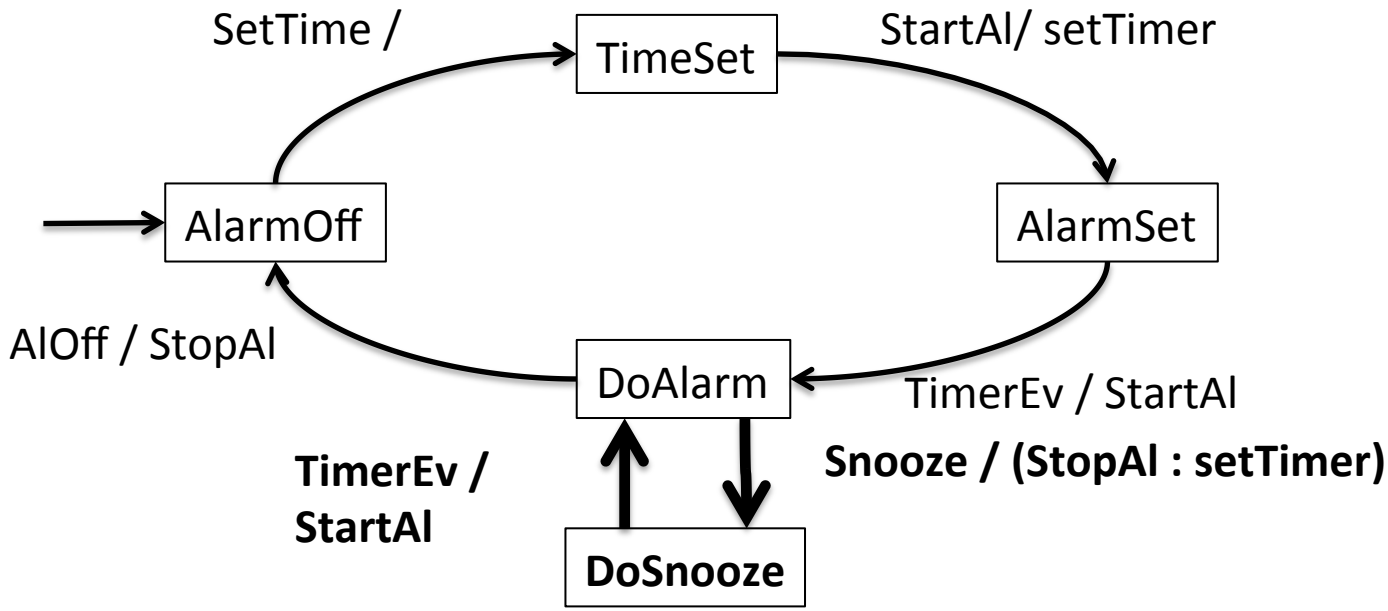
http://tapps-project.eu, Trusted Apps for open CPS
Picture: Energica Ego Superbike, ©EMC

Overview

- Extensions of state transition diagrams (SDs)
  - Used to add features to a SD
- Identify typical patterns of extensions / features
  - Capture extensions of SDs as refinement on traces
- Property preservation for typical patterns of properties
  - Extension patterns vs property patterns, case by case

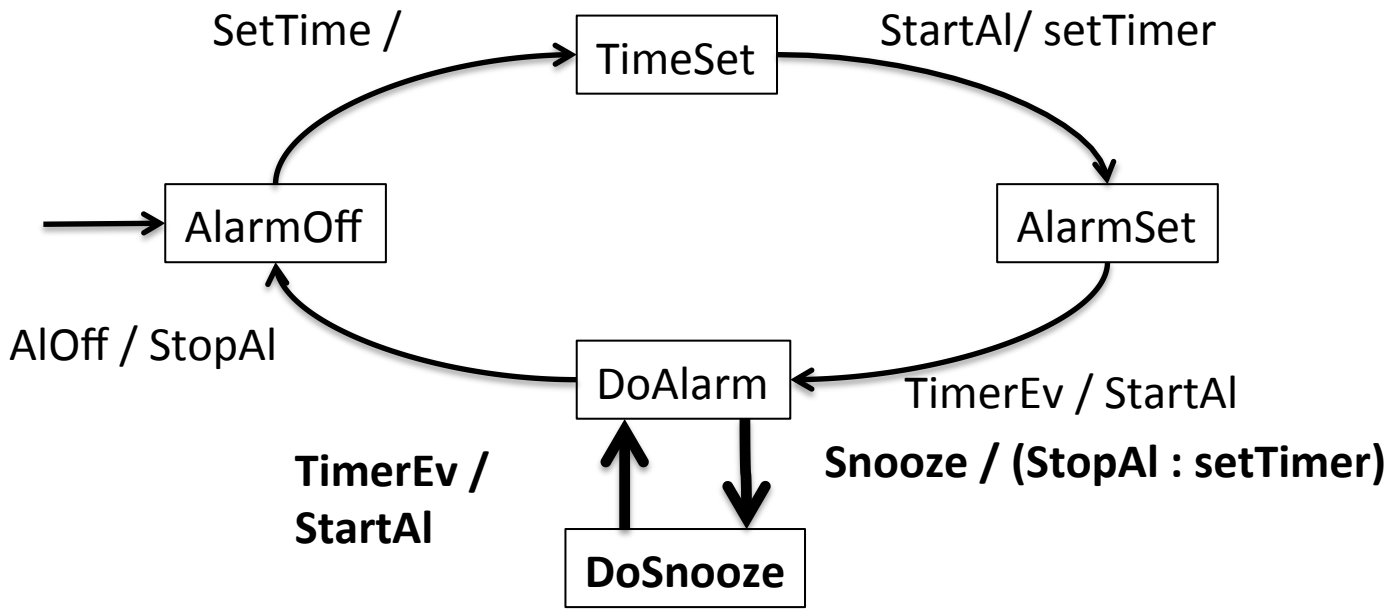
Extensions of State Transition Diagrams

- Extensions add new states and transitions to an existing SD
- Extensions are shown in bold



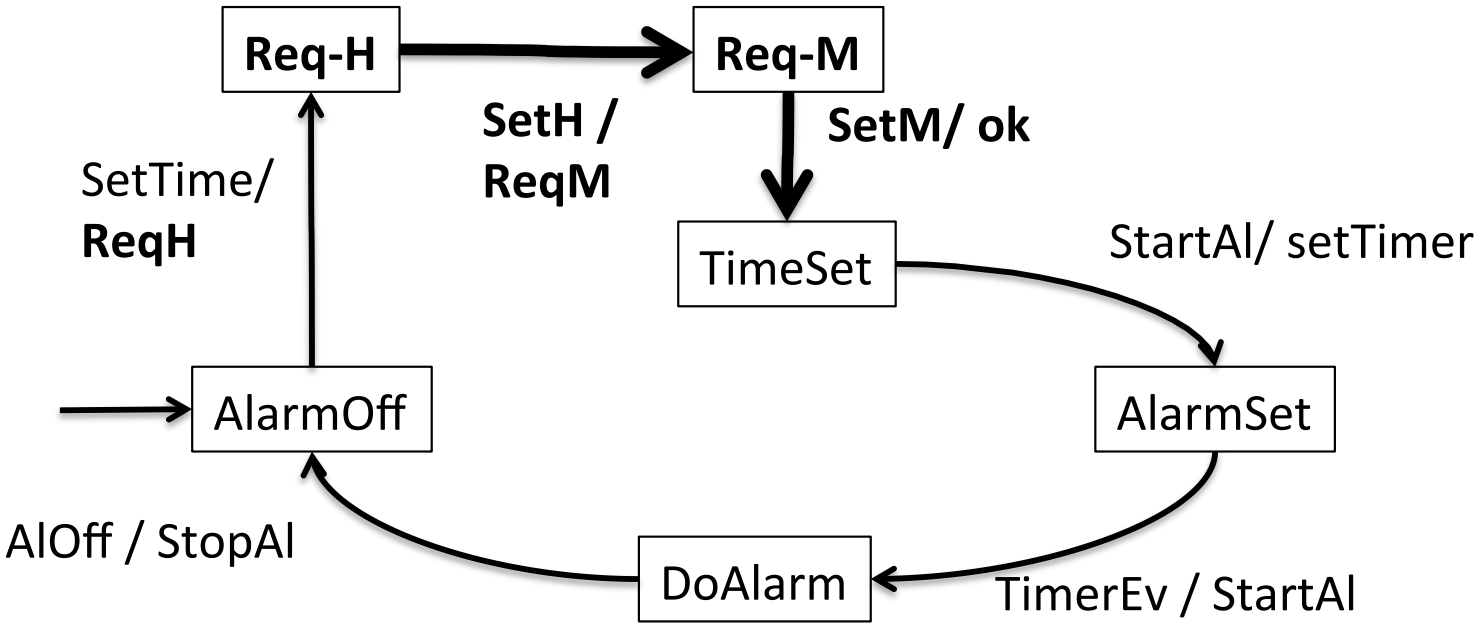
Pattern 1: Extension at a state

Extensions at a state add new behavior local to some state *s*.



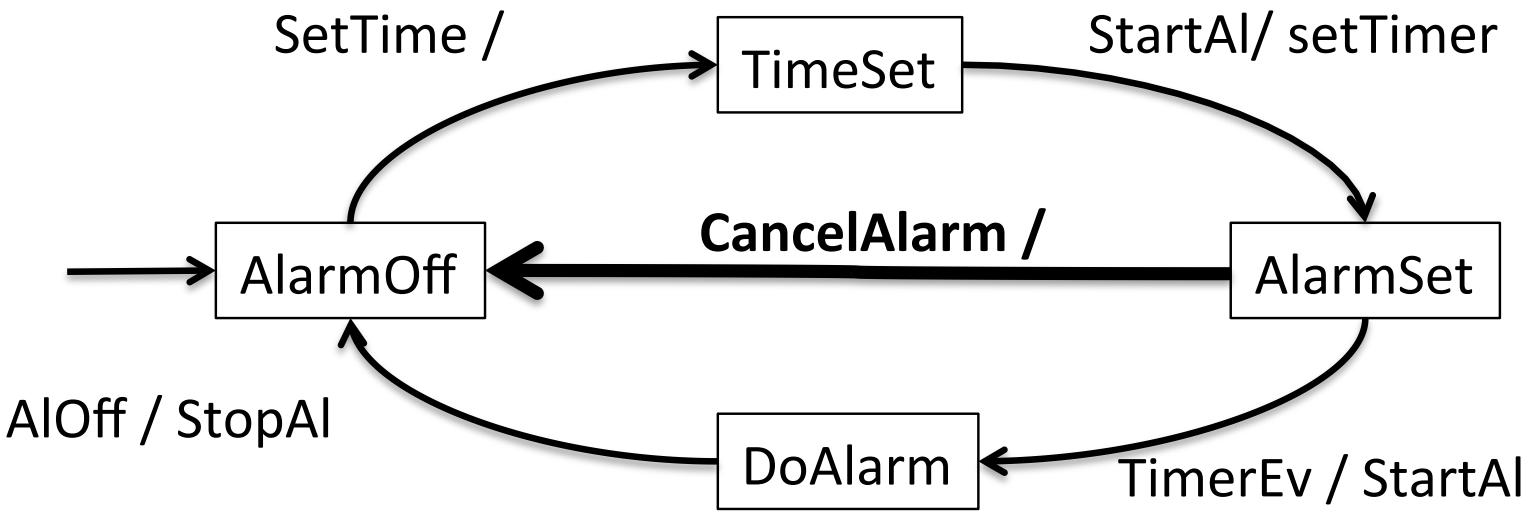
Pattern 2: Refining transitions

Refining transitions details the behavior of a transition.



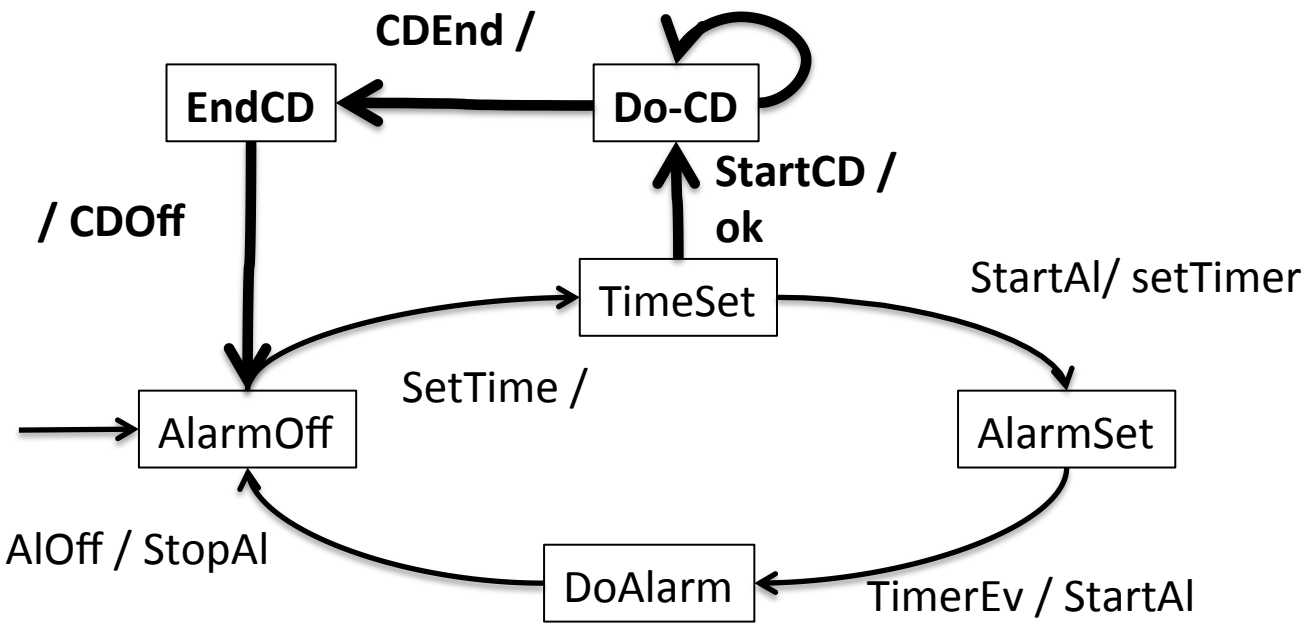
Pattern 3: Adding Failure Cases

Adding failure cases by transitions which lead back to a initial state.



Pattern 4: Other, arbitrary Functionality

Additional, arbitrary functionality initiated by a new event.



Elimination-based Refinement

- Existing refinement relations not sufficient for above example
  - E.g. only elimination of new events or cannot handle divergence
- New idea: Elimination-based refinements
  - Eliminate new behavior based on entry and exit events
  - Does not create a typical bisimulation

$$
\begin{array}{lllllllll}
(\mathit{SetTime} & : & \mathit{StartAl} & : & \mathit{TimerEvent} & : & \underline{\mathit{Snooze}} & : & \underline{\mathit{TimerEvent}} \\
\_ & : & \mathit{setTimer} & : & \mathit{StartAlarm} & : & \underline{\underline{(\mathit{StopAlarm}}}_{\mathit{:\ setTimer)}} & : & \underline{\mathit{StartAlarm}}
\end{array}
\quad : \mathit{AlOff},\ \ : \mathit{StopAlarm})
$$

Typical Patterns of Behavior Properties (Dwyer, 1999)

- **Simple response** of the form $\Box(P \implies \Diamond Q)$, where some event specified by P is always followed by some events specified by Q
- **Simple reachability**, i.e. $\Diamond Q$. This is called existence in Dwyer 1999.
- **Invariants** which hold for all transitions in an execution.
- **Compatibility** means identical behavior of the old and new SDs, assuming only the input events of the original SD.

Property Preservation for Extension/Feature Patterns

	Simple response (liveness)	**Simple reachability (liveness)**	**Invariants on events (safety)**	**Compatibility**
Extension at States	In case of strong refinement		If holds for extension	+
Transition Refinement	In case of strong refinement		If holds for extension	o
Adding Failures	o	+	+	+
Additional Functionality	o	o	If holds for extension	+

Note: Strong refinement means that the extension must terminate (for considered inputs)

Conclusions

- Refinement patterns in an integrated framework for state transition diagrams.
- Novel concepts of elimination-based refinement for these four classes of extensions based on assume/guarantee specifications.
- Analyzed property preservation for typical classes of safety and liveness properties wrt extension patterns.