Towards Complex Actions for Complex Event Processing

Steffen Hausmann and François Bry

Institute for Informatics, University of Munich

July 2, 2013
Motivating Use Case

Fire in a metro station:
- smoke is the most dangerous thread for passengers
- evacuation needs to be carried out quickly (~ within 15 min)
- operators need to focus on important decisions

Emergency management requirements:
- reliable situation assessment based on simulations
- execution of predefined dynamic reactions
- actions suited for external actuators
Particularities of External Actions

External actions significantly differ from internal actions:

- executed by isolated external actuators
- may fail to achieve their goal
- subject to runtime effects
- often irreversible
FOR
  smoke-extraction{ var Area }
DO
  compound {
    action a: open-fire-dampers{ var Area },
    action b: activate-ventilators{ var Area }
  } where { succ(a) <= init(b) }
  hence { succ(b)-init(a) <= 30 sec }
  succeeds on {
    event e: smoke{ area{var Area}, amount{var C} }
    where { var C < 0.2, end(e)-init(b) < 60 sec }
  }
END
FOR
  smoke-extraction{ var Area }
DO
  compound {
    action a: open-fire-dampers{ var Area },
    action b: activate-ventilators{ var Area }
  } where { succ(a) <= init(b) }
  hence { succ(b)-init(a) <= 30 sec }
  succeeds on {
    event e: smoke{ area{var Area}, amount{var C} }
    where { var C < 0.2, end(e)-init(b) < 60 sec }
  }
END
FOR
  smoke-extraction{ var Area }
DO
  compound {
    action a: open-fire-dampers{ var Area },
    action b: activate-ventilators{ var Area }
  } where { succ(a) <= init(b) }
  hence { succ(b)-init(a) <= 30 sec }
  succeeds on {
    event e: smoke{ area{var Area}, amount{var C} }
    where { var C < 0.2, end(e)-init(b) < 60 sec }
  }
END
Complex Actions in Dura

FOR
   smoke-extraction{ var Area }
DO
   compound {
      action a: open-fire-dampers{ var Area },
      action b: activate-ventilators{ var Area }
   } where { succ(a) <= init(b) }
   hence { succ(b)-init(a) <= 30 sec }
succeeds on {
      event e: smoke{ area{var Area}, amount{var C} }
      where { var C < 0.2, end(e)-init(b) < 60 sec }
   }
END
FOR
    smoke-extraction{ var Area }
DO
    compound {
        action a: open-fire-dampers{ var Area },
        action b: activate-ventilators{ var Area }
    } where { succ(a) <= init(b) }
    hence { succ(b)-init(a) <= 30 sec }
    succeeds on {
        event e: smoke{ area{var Area}, amount{var C} }
        where { var C < 0.2, end(e)-init(b) < 60 sec }
    }
END
There are four desirable dimensions of complex actions:

- action composition
- temporal dependencies
- execution result
- temporal assertions
There are four desirable dimensions of complex actions:

- action composition
- temporal dependencies
- execution result
- temporal assertions

Four orthogonal dimensions that should be kept separate.
Desirable properties of complex actions:

- **viability** execution according to temporal dependencies is feasible
- **fairness** each sub-action may be actually executed at runtime
- **entailment** the temporal assertions hold during runtime
Need for Static Temporal Analysis

Desirable properties of complex actions:

- **viability** execution according to temp. dependencies is feasible
- **fairness** each sub-action may be actually executed at runtime
- **entailment** the temporal assertions hold during runtime

Some complex actions do not fulfill these properties:

- ... where \{ \text{init}(a) = \text{init}(b) \} 
- ... where \{ \text{succ}(a) \leq \text{init}(b), \text{succ}(b) \leq \text{init}(a) \} 
- ... where \{ \text{succ}(a) \leq \text{init}(b) \} 
  hence \{ \text{succ}(b)-\text{init}(a) \leq 30 \text{ sec} \}
Formal Specification of Complex Actions:
- temporal dependencies: a conjunction of inequalities (C)
- temporal assertions: a formula built from inequalities (H)

Predict the execution of a complex action:
- predict runtime delays (scenario $\Delta$)
- determine how the action would be executed (trace $\tau_\Delta$)
- verify properties of the action for the predicted trace $\tau_\Delta$
Formal Specification of Complex Actions:
- temporal dependencies: a conjunction of inequalities (C)
- temporal assertions: a formula built from inequalities (H)

Predict the execution of a complex action:
- predict runtime delays (scenario $\Delta$)
- determine how the action would be executed (trace $\tau_\Delta$)
- verify properties of the action for the predicted trace $\tau_\Delta$

A complete analysis needs to consider all possible traces.
Definition (viability)
All atomic temporal dependencies have the form
\[ \text{variable} + \text{duration} \leq \text{identifier}_{\text{init}} \]

Definition (fairness)
For all sub-actions \( a \) there is a predicted trace \( \tau_\Delta: \tau_\Delta(a_{\text{init}}) < \infty \)

Definition (entailment)
For all predicted traces \( \tau_\Delta \) the variable free formula \( \tau_\Delta(H) \) holds
Definition (viability)

All atomic temporal dependencies have the form

\[ \text{variable} + \text{duration} \leq \text{identifier}_{\text{init}} \]

Definition (fairness)

For all sub-actions \( a \) there is a predicted trace \( \tau_{\Delta} : \tau_{\Delta}(a_{\text{init}}) < \infty \)

Definition (entailment)

For all predicted traces \( \tau_{\Delta} \) the variable free formula \( \tau_{\Delta}(H) \) holds

These properties are \textit{statically} decidable, that is, at compile time.
Basic ideas of the translation:

- actions → events for request, initiation, success and failure
- parameters and time of actions → payload of the events
- action invocations → declarative rules

```
FOR
  extract{ ... }
DO
  compound{
    action a: damper{ ... },
    action b: ventilator{ ... }
  } where {succ(a) <= init(b)}
END
```
Basic ideas of the translation:

- actions → events for request, initiation, success and failure
- parameters and time of actions → payload of the events
- action invocations → declarative rules

FOR
  extract{ ... }
DO
  compound{
    action a: damper{ ... },
    action b: ventilator{ ... }
  } where {succ(a) <= init(b)}
END
Translating Complex Actions to Deductive Rules

Basic ideas of the translation:

- actions → events for request, initiation, success and failure
- parameters and time of actions → payload of the events
- action invocations → declarative rules

FOR
  extract{ ... }
DO
  compound{
    action a: damper{ ... },
    action b: ventilator{ ... }
  } where {succ(a) <= init(b)}
END

DETECT
  ventilator$request{ payload{var P} }
ON
  and{
    event: extract$initiated{
      payload{var P}, id{var Id}
    },
    event: damper$success{ ref{var Id} }
  }
END
Translating Complex Actions to Deductive Rules

Basic ideas of the translation:
- actions → events for request, initiation, success and failure
- parameters and time of actions → payload of the events
- action invocations → declarative rules

```
FOR
    extract{ ... } DO
    compound{
        action a: damper{ ... },
        action b: ventilator{ ... }
    } where {succ(a) <= init(b)} END

DETECT
    ventilator$request{ payload{var P} }
    ON
    and{
        event: extract$initiated{
            payload{var P}, id{var Id}
        },
        event: damper$succeed{ ref{var Id}
    }
    END
```
Conclusions

Complex actions for emergency management:

- dynamic reactions that adopt to the ongoing situation
- applied to three emergency management use cases
  - metro
  - airport
  - power grid
- subject to a static temporal analysis
- suited for various domains
- transferable to other event processing languages
Towards Complex Actions for Complex Event Processing

Steffen Hausmann and François Bry

Institute for Informatics, University of Munich

July 2, 2013