Contents & Goals

Last Lecture:
• The basic causality model
• Ether

This Lecture:
• Educational Objectives:
  • Capabilities for following tasks/questions.
  • What does this State Machine mean? What happens if I inject this event?
  • Can you please model the following behaviour.
  • What is: Signal, Event, Ether, Transformer, Step, RTC.
• Content:
  • System Configuration, Transformer
  • Examples for transformer
  • Run-to-completion Step
  • Putting It All Together

Roadmap: Chronologically
(i) What do we (have to) cover?
  UML State Machine Diagrams
  Syntax.
(ii) Def.: Signature with signals.
(iii) Def.: Core state machine.
(iv) Map UML State Machine Diagrams to core state machines.
  Semantics: The Basic Causality Model
(v) Def.: Ether (aka. event pool)
(vi) Def.: System configuration.
(vii) Def.: Event.
(viii) Def.: Transformer.
(ix) Def.: Transitions system, computation.
(x) Transition relation induced by core state machine.
(xi) Def.: step, run-to-completion step.
(xii) Later: Hierarchical state machines.
except for $E$ is defined foreach $\sigma$. For a temporary association to access $C$, i.e. $a$ is a boolean value, $\sigma, \varepsilon$ is a pair which is a general representation of the possible choices.

Our $\sigma$ dom($\emptyset$) = $0 \cup \{\langle \cdot, \cdot \rangle\}$, and $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals. $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals.

Stability $\sigma, \varepsilon$ is preserved.

But: we'll later briefly discuss "discarding" of events.

Oftenseen minimal requirement $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals. In some cases, this is instantaneous and completely reliable while in others it may involve transmission delays of variable duration, loss of requests, medium, and numerous other factors. An operation to clear the ether for a given object, i.e. $\sigma, \varepsilon$ is a signature with signals. $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals.

Semantic Variation Points: $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals, $\sigma, \varepsilon$ is a signature with signals.
Special cases to u has been sent from E (resulting in an action language)

\[ \sigma, \epsilon \] is some system configuration of the corresponding signature.

Examples

\[ \langle \text{action} \rangle \] and \[ \langle \text{guard} \rangle \] are event signals (classes) of the transition.

Intuitively, \[ s \] is the effect of applying the transformer \[ \varepsilon \], \[ \sigma \] and which parameters it carries.

\[ \text{Introduce the (simplified) syntax of transition annotations:} \]

\[ \text{event ::=} \text{annot} \]

\[ \text{An example of the corresponding syntax:} \]

\[ \text{event} := \text{annot} \]

\[ \text{ExpressionLanguage} \]

\[ \text{• Identity} \]

\[ \text{Signals are instances of} \]

\[ \text{Events} \]

\[ \text{are} \]

\[ \text{instances of signals (classes)} \]

\[ \text{of the effect of applying the} \]

\[ \text{transformer} \]

\[ \text{and which parameters it carries.} \]

Why Transformers?

(ii) Explain how to choose transitions depending on descriptions of behaviour, and which parameters it carries.

(i) Introduce the (simplified) syntax of transition annotations:

\[ \text{event ::=} \text{annot} \]

\[ \text{An example of the corresponding syntax:} \]

\[ \text{event} := \text{annot} \]

\[ \text{ExpressionLanguage} \]

\[ \text{• Identity} \]

\[ \text{Signals are instances of} \]

\[ \text{Events} \]

\[ \text{are} \]

\[ \text{instances of signals (classes)} \]

\[ \text{of the effect of applying the} \]

\[ \text{transformer} \]

\[ \text{and which parameters it carries.} \]
\[ \mathcal{E} \text{ not defined.} \]

\[ i \cdot \mathcal{E} \]

\[ \mathcal{E} \text{ remains unchanged} \]

\[ \mathcal{E} \text{ update} \]

\[ \mathcal{E} \text{ semantics} \]

\[ \mathcal{E} \text{ obeys visibility and navigability} \]

\[ \mathcal{E} \text{ updates local state — boring} \]

Assuming \( \mathcal{E} \text{ defined} \) (for instance because of dangling reference navigation or division by zero), we store it as a partial function to handle undefined values during runtime. If \( \mathcal{E} \text{ undefined} \) (perhaps because of multiple objects or division by zero), we use a special handling mechanism.

In the following, we assume \( \mathcal{E} \text{ given} \) and that we're interested in the pre-images which map to \( \mathcal{E} \).

Transformers as Abstract Actions!
References

