Institutions for UML State Machines

Martin Glauer

December 10, 2014

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

UML State Machines

UML

- **U**nified **M**odelling **L**anguage
- Standard managed by the **Object Management Group** (OMG)
- Structural and behavioural modelling

UML State Machines

- Behavioural modelling
- Similar to finite automatons (but way more powerful)
- Hard to check (e.g. consistency)

UML State Machines - States and Transitions

States

- Manly labelled boxes

Transitions

- **Source/Target**
- **Trigger**: Event that must be first in event pool
- **Guard**: Condition that must be satisfied
- **Effect**: Actions caused by transition

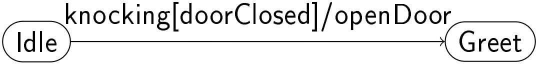
UML State Machines - States and Transitions

States

- Manly labelled boxes

Transitions

- **Source/Target**
- **Trigger**: Event that must be first in event pool
- **Guard**: Condition that must be satisfied
- **Effect**: Actions caused by transition



Institutions

- *Sign* - Category of signatures
- $sen : Sign \rightarrow Set$ - Sentences given a signature Σ, translates sentences along a morphism
- $Mod : Sign^{op} \rightarrow cat$ - category of models given an signature Σ, translates models countering a morphism
- $\models_{\Sigma} \ \subseteq |Mod(\Sigma)| \times sen(\Sigma)$ - satisfaction relation
- **Satisfaction condition**:
  $M' \models_{\Sigma'} Sen(\sigma)(\gamma)$ iff $Mod(\sigma)(M) \models_{\Sigma} \gamma$

Institution of Actions

$H = (A_H, M_H, V_H)$

- A_H - Set of actions
- M_H - Set of messages
- V_H - Set of variables

Config. Transition: $\Omega \subseteq |\Omega| \times (A_H \times \wp(M_H)) \times |\Omega|$

- $|\Omega| = (V_H \to \mathrm{Val})$
- $\omega \xrightarrow[\Omega]{a,\overline{m}} \omega'$: Transition from config ω to ω' on action a whilst messages $\overline{m}$ are emitted
- e.g. $\{i \mapsto 0, j \mapsto 0, ...\} \xrightarrow[\Omega]{i \leftarrow i+1,\ \emptyset} \{i \mapsto 1, j \mapsto 0, ...\}$

Signature - Machines

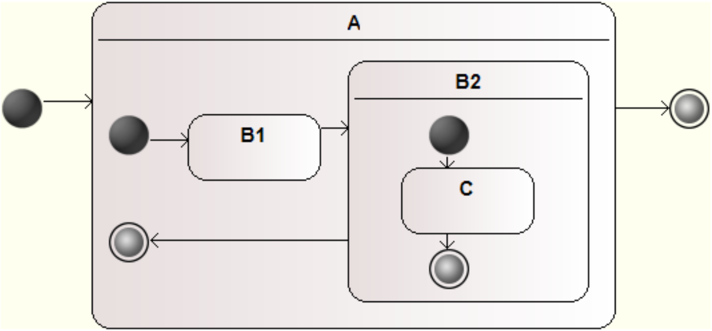
$\Sigma = (E_\Sigma, F_\Sigma, S_\Sigma)$

- E_Σ - Set of events
- F_Σ - Set of completion events
- S_Σ - Set of states

State Machine Signature

- Where do we start? Where can we go?
- $I_\Theta \in \wp(V_H \rightarrow \mathrm{Val}) \times S_\Sigma$ - initial configuration
- $(\omega, p :: \overline{p}, s) \xrightarrow[\Delta_\Theta]{\overline{m} \setminus E_\Sigma} (\omega', \overline{p} \lhd ((\overline{m} \cap E_\Sigma) \cup \overline{f}), s')$
  - if $\exists s \xrightarrow[T]{p[g]/a,\overline{f}} s'$ such that $\omega \models g$ and $\omega \xrightarrow[\Omega]{a,\overline{m}} \omega'$
- if there is no such transition: drop event p from event pool

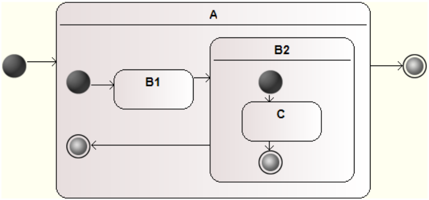
State Machines - Nested States



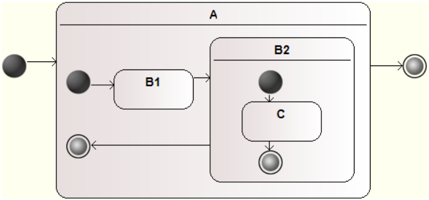
State Machine Signature - Hier. States

- Where do we start? Where can we go?
- $I_\Theta \in \wp(V_H \rightarrow \mathrm{Val}) \times [S_\Sigma]$ - initial configuration
- $(\omega, p :: \overline{p}, s :: L) \xrightarrow[\Delta_\Theta]{\overline{m} \setminus E_\Sigma} (\omega', \overline{p} \lhd ((\overline{m} \cap E_\Sigma) \cup \overline{f}), L')$
  - if $\exists s \xrightarrow[T]{p[g]/a,\overline{f}} s'$ such that $\omega \models g$ and $\omega \xrightarrow[\Omega]{a,\overline{m}} \omega'$
  - if s' is a non-final state: $L' = \mathrm{childRec}(s') : (s' :: L)$
    if s' is a final state: $L' = L$
- if there is no such transition: drop event p from event pool

State Machines - Nested States

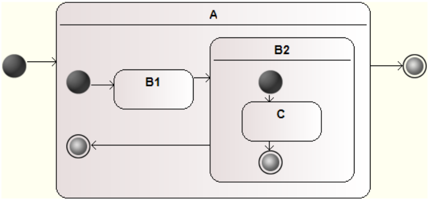


State Machines - Nested States



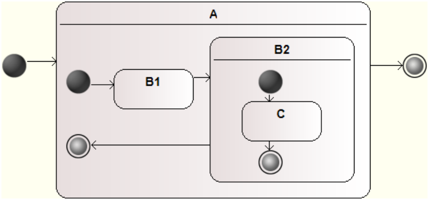
- $[B1, A] \rightarrow$

State Machines - Nested States



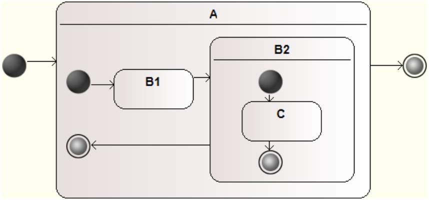
- $[B1, A] \rightarrow [C, B2, A] \rightarrow$

State Machines - Nested States



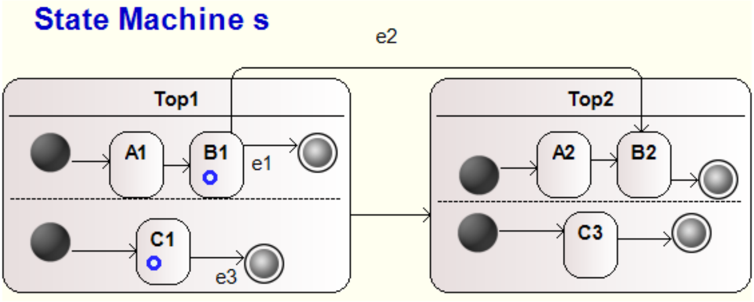
- $[B1, A] \to [C, B2, A] \to [B2, A] \to$

State Machines - Nested States



- $[B1, A] \to [C, B2, A] \to [B2, A] \to [A] \to []$

State Machines - Forks

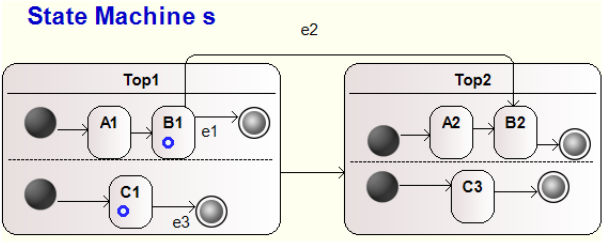


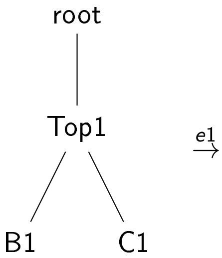
State Machine Signature - Forks

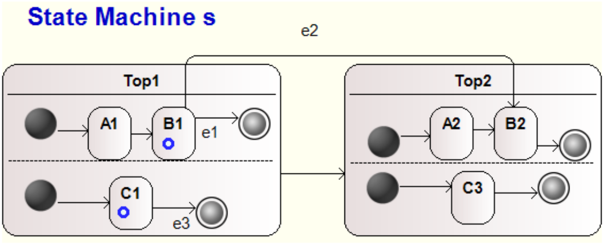
- Where do we start? Where can we go?
- $I_\Theta \in \wp(V_H \to \mathrm{Val}) \times \mathit{Tree}(S_\Sigma)$ - initial configuration
- $(\omega, p :: \overline{p}, T) \xrightarrow[\Delta_\Theta]{\overline{m} \backslash E_\Sigma} (\omega', \overline{p} \lhd ((\overline{m} \cap E_\Sigma) \cup \overline{f}), T')$
  - if $\exists s \in \mathit{leaves}(T) : s \xrightarrow[T]{p[g]/a, \overline{f}} s'$ such that $\omega \models g$ and $\omega \xrightarrow[\Omega]{a, \overline{m}} \omega'$ and $T \xrightarrow{s, s'} T'$
  - $T \xrightarrow{s, s'} T'$ expresses that T' originates from T by:
    - if s is no final state: s is replaced by $\mathit{ForwardTree}(s')$
    - if s' is a final state: s is dropped
  - if there is no such transition: drop event p from event pool

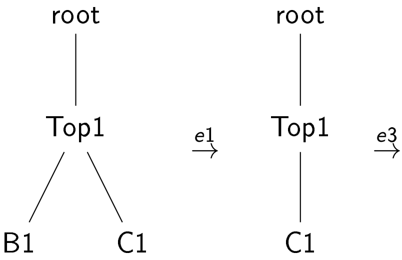
ForwardTree

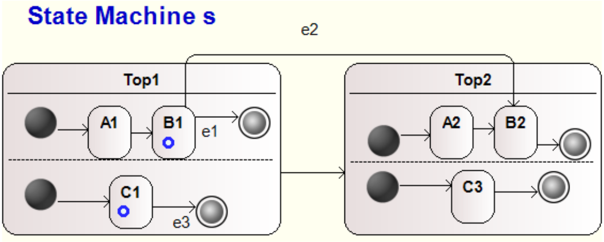
```
ForwardTree(s):
    if s is simple State:
        return Tree(s)
    else: #s is composite state
        T = Tree(s)
        T.children =
          [ForwardTree(c) for c in subregions(s)]
        return T
```

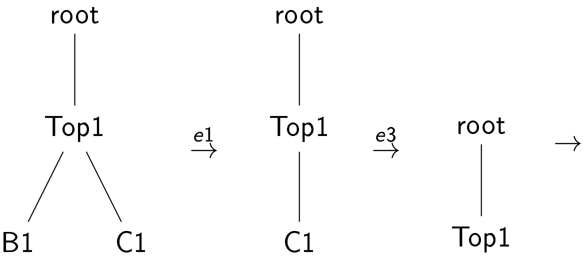

State Machine s
e2
Top1
A1
B1
e1
C1
e3
Top2
A2
B2
C3

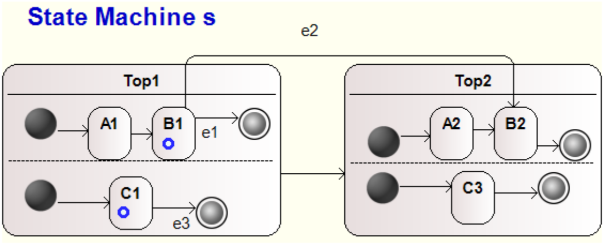

root
Top1
B1
C1
$\xrightarrow{e1}$

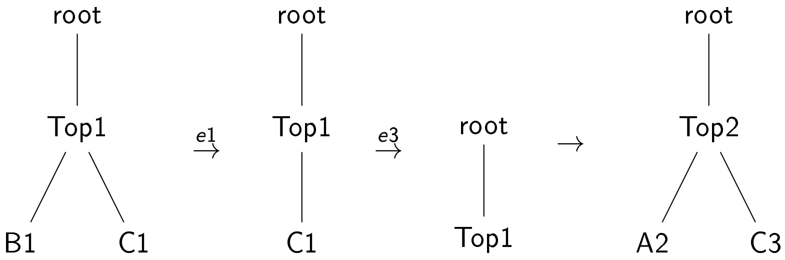

State Machine s
e2
Top1
A1
B1
e1
C1
e3
Top2
A2
B2
C3

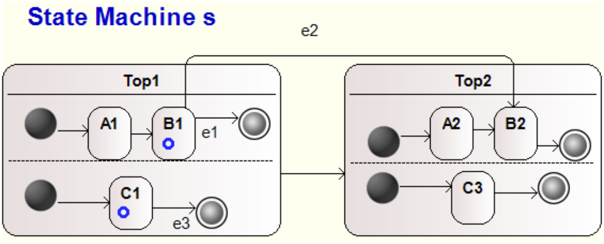

root
Top1
B1
C1
e1
root
Top1
C1
e3

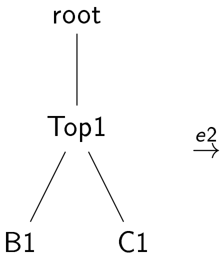

State Machine s
e2
Top1
A1
B1
e1
C1
e3
Top2
A2
B2
C3

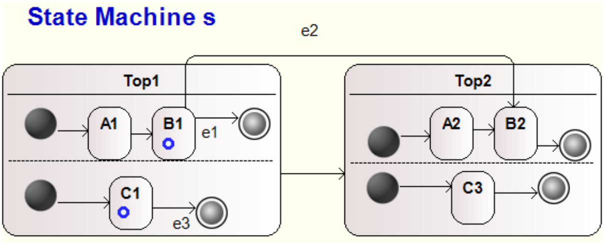

root
Top1
B1
C1
e1
root
Top1
C1
e3
root
Top1

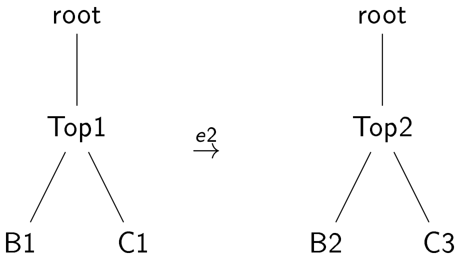

State Machine s
e2
Top1
A1
B1
e1
C1
e3
Top2
A2
B2
C3


root
Top1
B1
C1
e1
root
Top1
C1
e3
root
Top1
root
Top2
A2
C3


State Machine s
e2
Top1
A1
B1
e1
C1
e3
Top2
A2
B2
C3


root
Top1
B1
C1
e2


State Machine s
e2
Top1
A1
B1
e1
C1
e3
Top2
A2
B2
C3


root
Top1
B1
C1
e2
root
Top2
B2
C3

State Machine Signature - Forks

- Where do we start? Where can we go?
- $I_\Theta \in \wp(V_H \to \mathrm{Val}) \times \mathit{Tree}(S_\Sigma)$ - initial configuration
- $(\omega, p :: \overline{p}, T) \xrightarrow[\Delta_\Theta]{\overline{m} \setminus E_\Sigma} (\omega', \overline{p} \lhd ((\overline{m} \cup E_\Sigma) \cup \overline{f}), T')$
  - if $\exists s \in T : s \xrightarrow[T]{p[g]/a,\overline{f}} s'$ such that $\omega \models g$ and $\omega \xrightarrow[\Omega]{a,\overline{m}} \omega'$ and $T \xrightarrow{s,s'} T'$
  - $T \xrightarrow{s,s'} T'$ expresses that T' originates from T by:
    - if s is no final state: The subtree of $lca(s, s')$ containing s is replaced by the tree containing s'
    - if s' is a final state: s is dropped
  - if there is no such transition: drop event p from event pool

Conclusion

- State Machines can be formalised as institutions
- Description of composite states by trees
- "Path to target" hard to describe
- There are other pseudo states (e.g. histories)

Institutions for UML State Machines

Martin Glauer

December 10, 2014

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INF
FAKULTÄT FÜR
INFORMATIK