

Komponensalapú reaktív rendszerek lépésenként vezérelhető szimulációja precíz formális szemantika szerint

Szkupien Péter



Konzulens: Dr. Molnár Vince



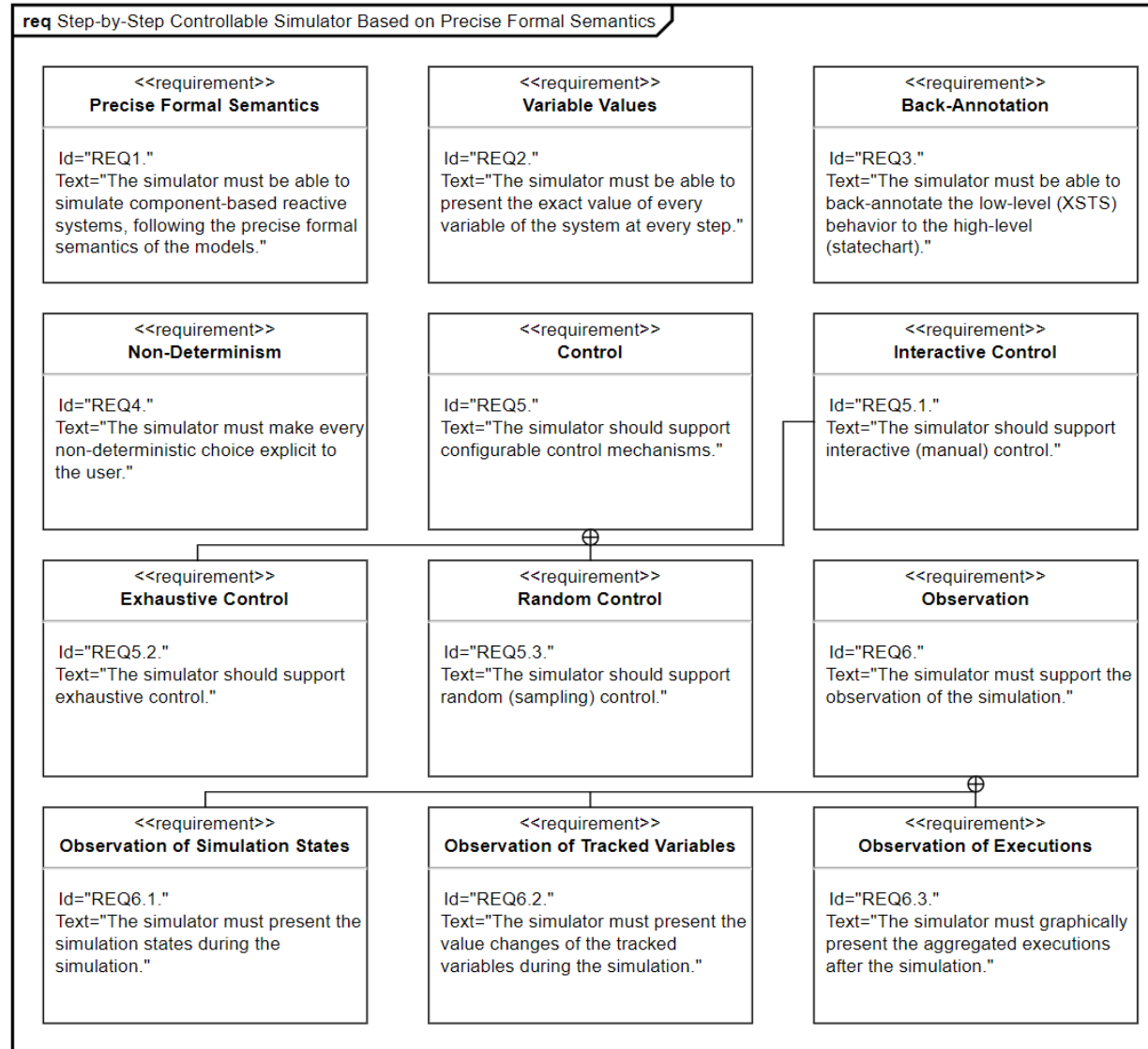
Budapest University of Technology and Economics
Department of Measurement and Information Systems
ftsrg Research Group



Bevezetés

- Modellalapú fejlesztés: **végrehajtható modellek**
- Komponensalapú reaktív rendszerek: **állapotgépek**
- Szimuláció vezérlése: **nemdeterminizmus**
- **Gamma**: állapotgép → Extended Symbolic Transition System (XSTS) 
- **Theta**: XSTS modellellenőrzés 

Szimulátor: követelmények



Precíz formális szemantika

- **REQ1.** *The simulator must be able to simulate component-based reactive systems, following the precise formal semantics of the models.*
 - Gamma \rightarrow XSTS modelltranszformáció
 - Theta XSTS modellellenőrző újrafelhasználása
 - Szimuláció modellellenőrzővel

Változó értékek

- **REQ2.** *The simulator must be able to present the exact value of every variable of the system at every step.*
 - Hatékonyság nem kritikus
 - Absztrakció mellőzése

Visszavetítés

- **REQ3.** *The simulator must be able to back-annotate the low-level (XSTS) behavior to the high-level (statechart).*
 - Gamma-szintű log utasítások (tetszőleges string)
 - Log utasítások leképezése XSTS változókra és értékadásokra
 - Értékváltozások megfigyelése

Nemdeterminizmus

- **REQ4.** *The simulator must make every non-deterministic choice explicit to the user.*
 - Teljes vezérelhetőség
 - *XSTS* tranzíciók feldarabolása

XSTS feldarabolás

- Tranzíciók feldarabolása *belső nemdeterminizmus* mentén
 - Choice
 - Parallel
 - Havoc
 - (Conditional)
- Modelltranszformáció
 - Nemdeterminizmus: *belső* → *külső*
 - Szemantika megtartása
 - Formalizált szabályok: 4.3 *Splitting Rules* fejezet
- Implementáció: Gamma utófeldolgozás

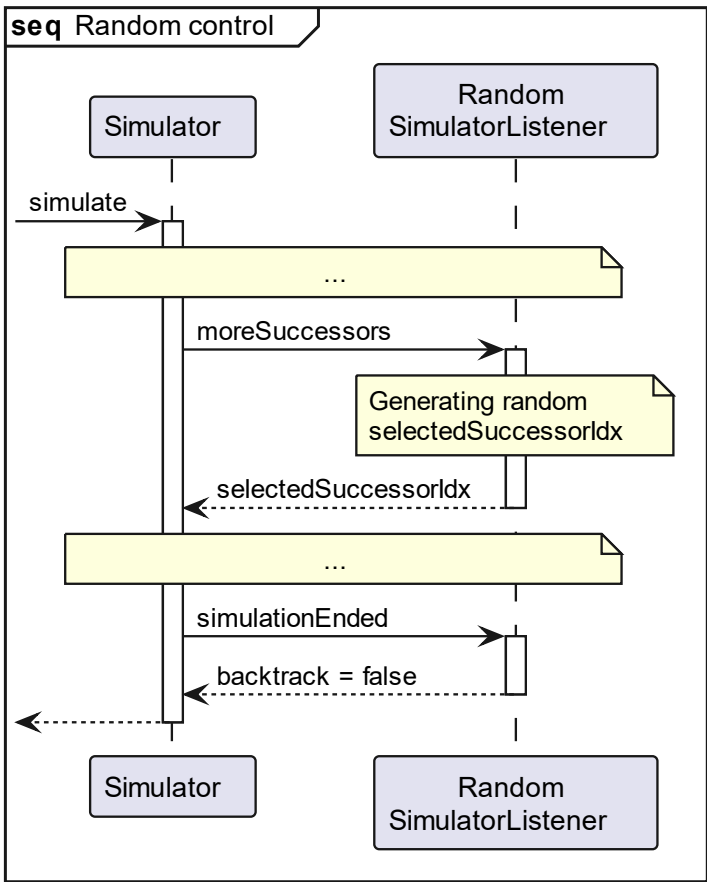
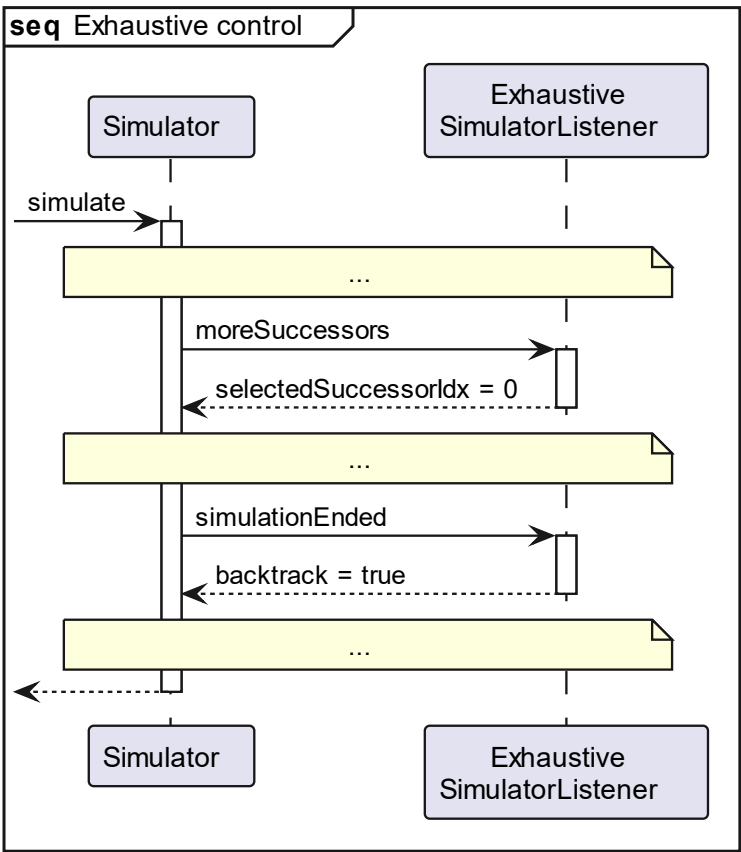
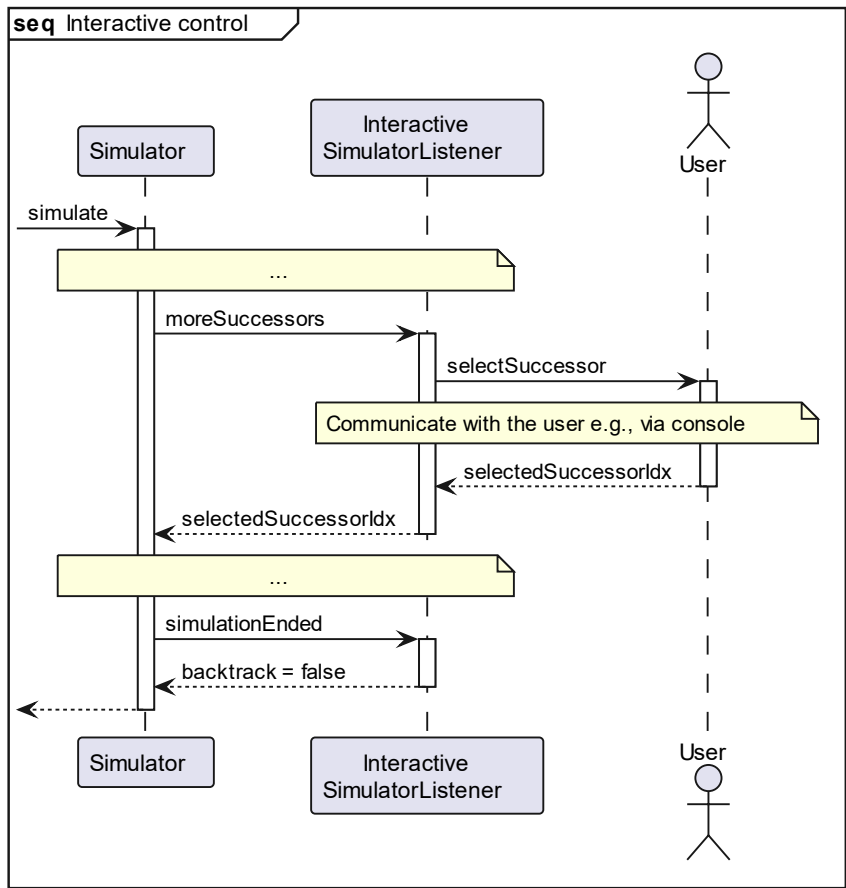


P. Szakupien, V. Molnár:
The Effect of Transition
Granularity in the Model
Checking of Reactive
Systems

Vezérlés

- **REQ5.** *The simulator should support configurable control mechanisms.*
 - **REQ5.1.** *The simulator should support interactive (manual) control.*
 - Felhasználó
 - **REQ5.2.** *The simulator should support exhaustive control.*
 - Minden lehetőség kimerítő bejárása
 - **REQ5.3.** *The simulator should support random (sampling) control.*
 - Nem kimerítő bejárás

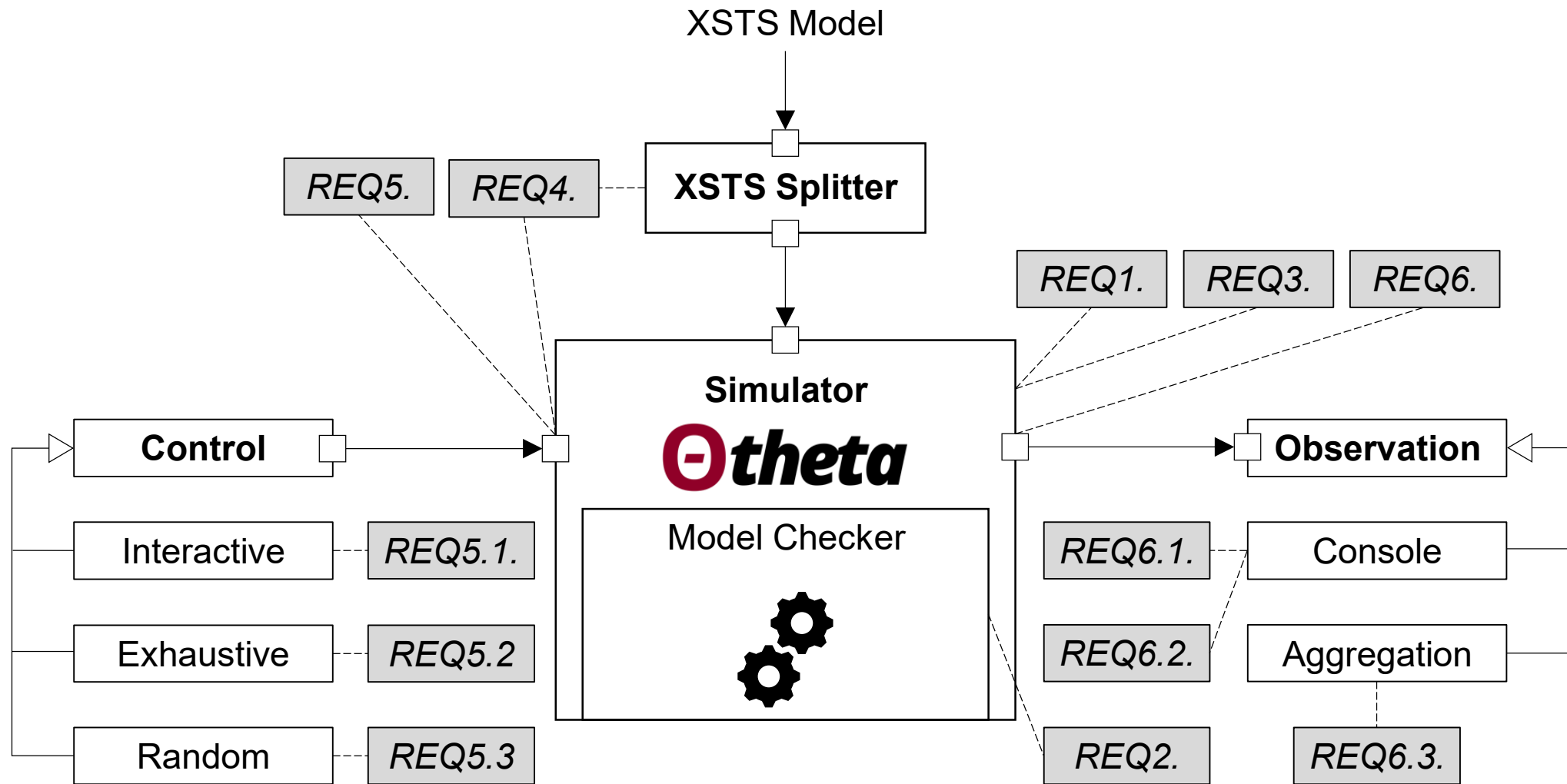
Vezérlés



Megfigyelés

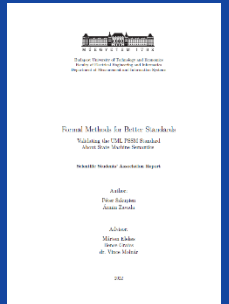
- **REQ6.** *The simulator must support the observation of the simulation.*
 - **REQ6.1.** *The simulator must present the simulation states during the simulation.*
 - Állapot: változó értékek
 - **REQ6.2.** *The simulator must present the value changes of the tracked variables during the simulation.*
 - Követett változók: értékváltozások
 - **REQ6.3.** *The simulator must graphically present the aggregated executions after the simulation.*
 - Végrehajtások: aggregált grafikus reprezentáció

Szimulátor: architektúra



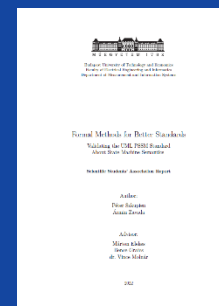
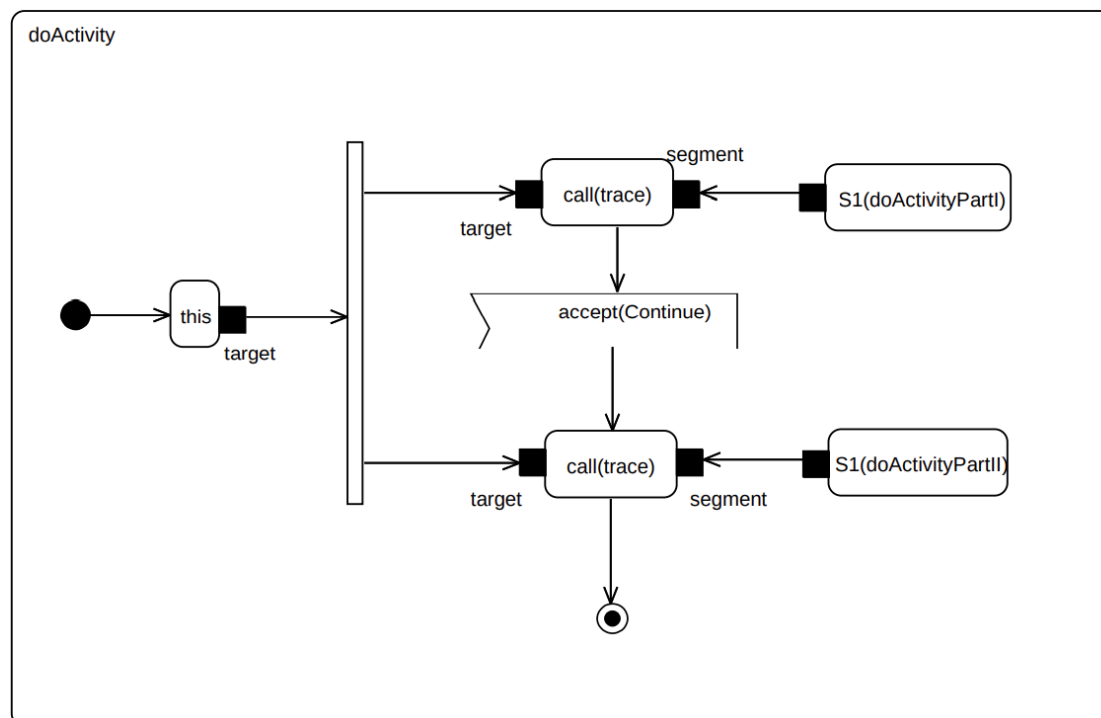
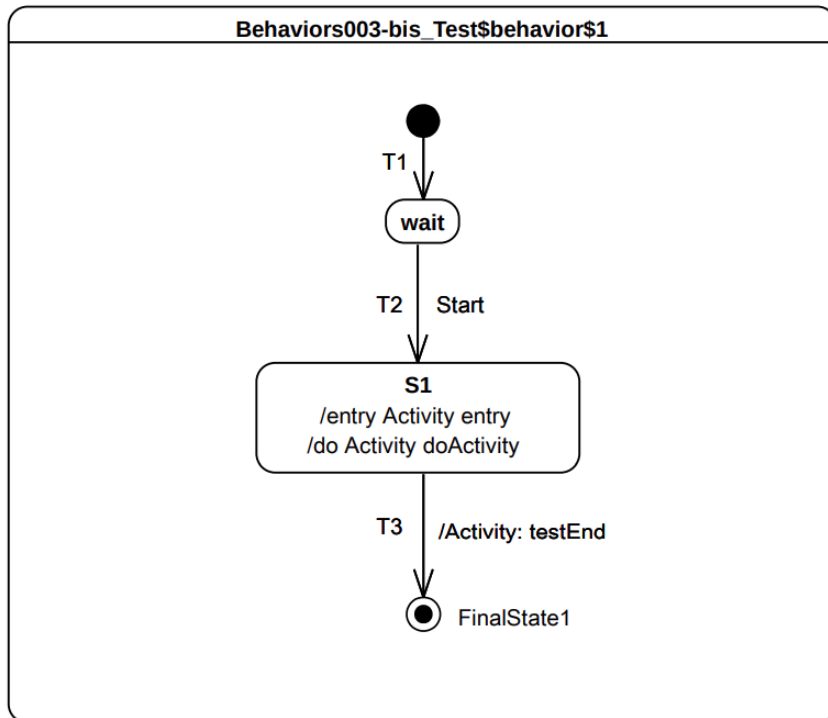
Példa

- UML PSSM szabvány vizsgálata
- PSSM teszt modellek → Gamma → XSTS
- Kimerítő szimuláció: lehetséges lefutások vizsgálata
- Összehasonlítás a szabvánnyal
 - *Feltárt hibák!*



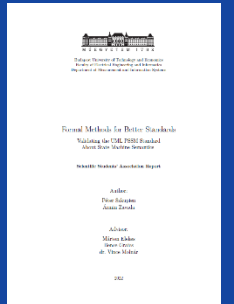
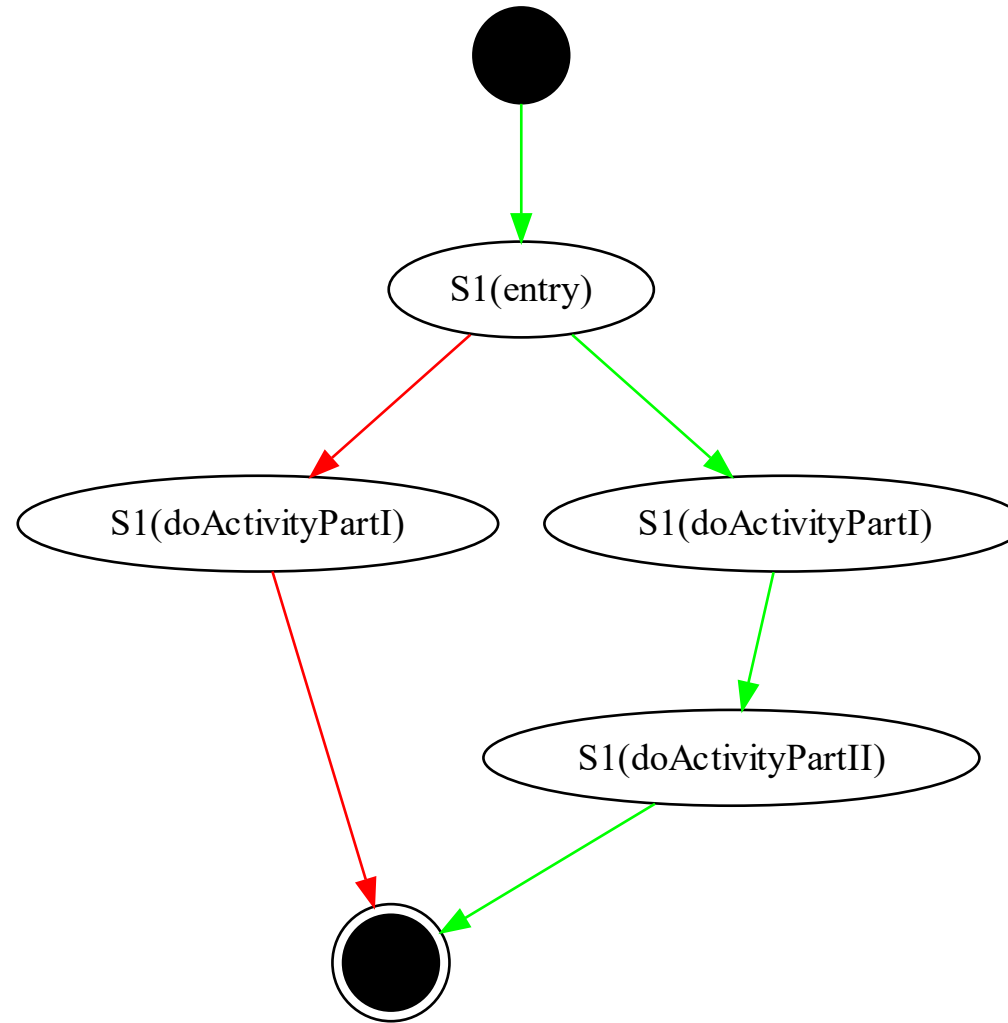
P. Szekupien, Á. Zawada:
Formal Methods for
Better Standards:
Validating the UML PSSM
Standard About State
Machine Semantics

Példa



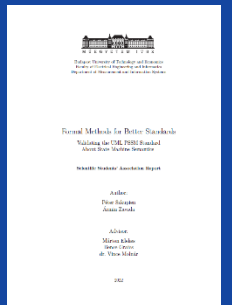
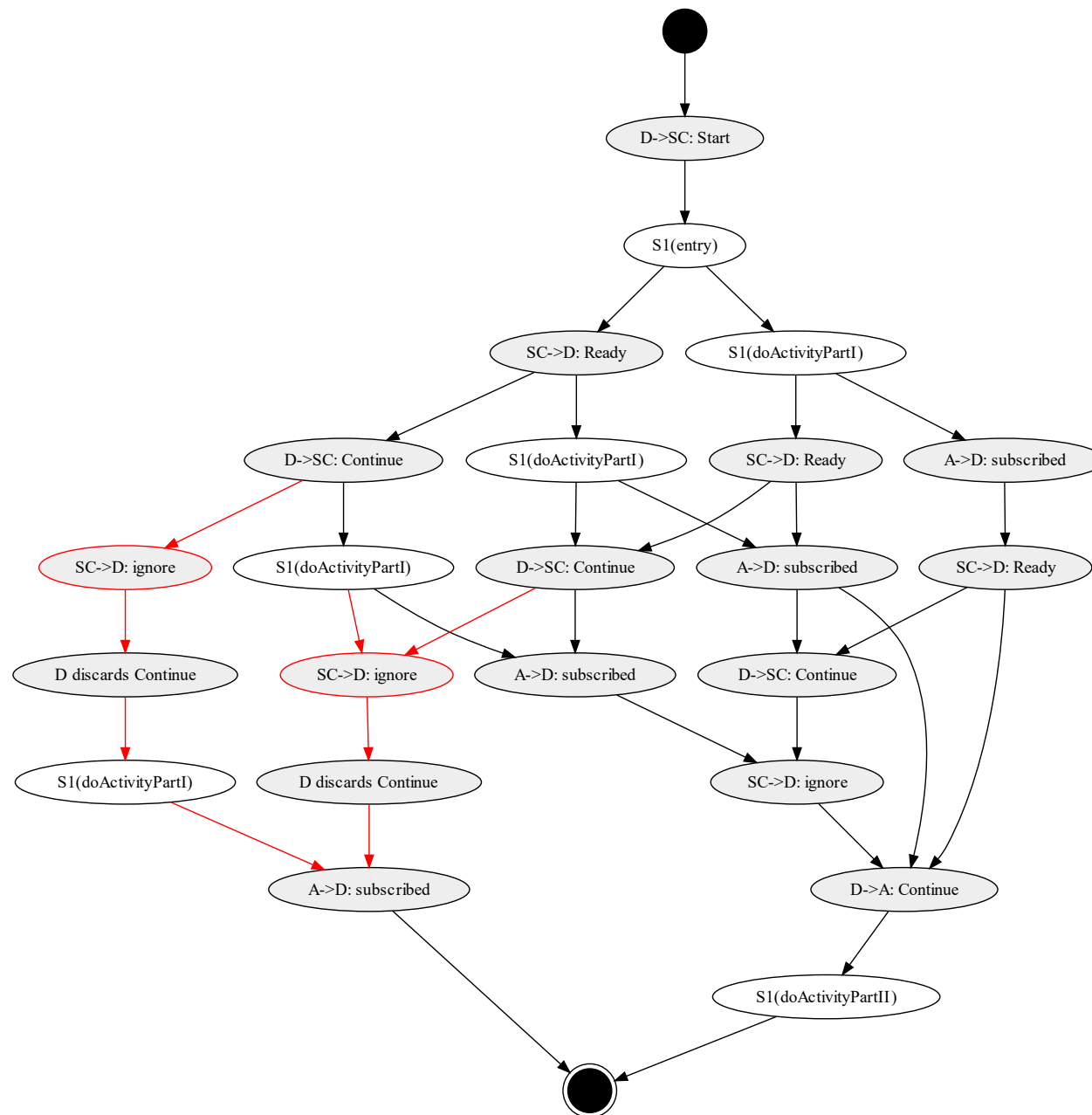
P. Szkupien, Á. Zavada:
Formal Methods for
Better Standards:
Validating the UML PSSM
Standard About State
Machine Semantics

Példa



P. Szkupien, Á. Zavada:
Formal Methods for
Better Standards:
Validating the UML PSSM
Standard About State
Machine Semantics

Példa



P. Szkupien, Á. Zavada:
Formal Methods for
Better Standards:
Validating the UML PSSM
Standard About State
Machine Semantics

Továbbfejlesztési lehetőségek

- Visszavetítés továbbfejlesztése
- Általános kifejezések értékváltozásainak követése
- Szimulációk mentése, visszatöltése
- Felhasználói felület továbbfejlesztése

Összefoglalás

- Kontextus, háttérismeretek
- Szimulátorral szemben támasztott követelmények definiálása
- XSTS feldarabolás formalizálása, implementálása
- Szimulációs keretrendszer megtervezése, implementálása
- Használhatóság demonstrálása
- Továbbfejlesztési lehetőségek

Bírálóí kérdések

[illegible][illegible]

Dr. Pandy Gergely

1. kérdés

A 2.2.3.1 szakasz *"Multiple fireable transitions: In real-life engineering work, especially with the growing complexity of state hierarchies and guard expressions, statecharts are usually not deterministic and fully defined."* állítása nehezen tűnik védhetőnek. Noha egy külső eseményre adott válasz egy sok komponensből álló rendszerben valóban tekinthető nemdeterminisztikusnak, ez csak abból ered, hogy a sok komponens állapota nincs szigorú szinkronban – egy konkrét állapotgéppel modellezett komponens modelljében a nemdeterminizmus minden bizonnyal hiba. **Milyen valós biztonságkritikus rendszerben tartja elképzelhetőnek, hogy a fejlesztők szándékosan nemdeterminisztikus viselkedést modelleznek egy állapotgépen belül?**

1. kérdés

[...] Milyen valós biztonságkritikus rendszerben tartja elképzelhetőnek, hogy a fejlesztők szándékosan nemdeterminisztikus viselkedést modelleznek egy állapotgépen belül?

- *Párhuzamos régiók* ütemezése
- *Környezetet/felhasználót* leíró modell
- *Technikailag* más nemdeterminizmus is lehet (pl. tranzíciók azonos triggerrel és nem kizáró őrfeltételekkel), de ezek biztonságkritikus kontextusban valóban hibák

2. kérdés

A 2.2.4.1 szakaszban bevezetett szemantika az UML összetett átmeneteit "operation"-ök alkotta szekvenciák, elágazások, párhuzamos működés, stb. struktúrájaként írja le, ahol az értékadások kb. az átmenetekhez rendelt aktivitásoknak, az "assumption"-nek nevezett mellékhatásmentes kifejezések kiértékelése pedig az őrfeltételeknek felelnek meg. Hogyan kezeli a szemantika azt a helyzetet, ha egy (akár elágazásmentes) átmenet láncon az első elemi átmenet (operation) őrfeltétele (assumption) teljesül, de a hozzá tartozó aktivitás (assignment) úgy módosítja az állapotteret (változók értékét), hogy a láncban következő átmenet őrfeltétele (assumption) már nem teljesül (akár úgy, hogy az átmenetek közötti vertex egy junction pszeudoállapot) – hogyan kerüli el a szemantika azt, hogy ilyen helyzetben ne álljon elő egy félig végrehajtott átmenet? Ha ezt a kérdést még egy modellellenőrzési probléma keretében lehet is kezelni, mit tudunk kezdeni azzal, ha egy valós szoftverben a modellezett elemi átmenethez rendelt aktivitások módosítják a környezetet, vagyis nem vonhatók vissza? A kérdésről a 10. oldal alján ez szerepel: *"Note that assumptions may cause any composite operation to yield an empty set as the set of successor states. This allows us to use the choice operation as a guarded branching operator, ruling out branches where an assumption fails by yielding an empty set as the result of that branch. In this work, we make the following assumptions, which can be easily guaranteed by simple pre-processing."* – mi az az "egyszerű előfeldolgozás", ami garantálja, hogy egy tetszőleges bonyolult elemi átmenet struktúra elkezdése valamilyen kiindulási állapotból biztosan nem okoz olyan változást a változók kiértékelésében, hogy a lánc egy későbbi "assumption"-ja ne teljesüljön?

2. kérdés

[...] hogyan kerül el a szemantika azt, hogy ilyen helyzetben ne álljon elő egy félig végrehajtott átmenet? Ha ezt a kérdést még egy modellellenőrzési probléma keretében lehet is kezelni, mit tudunk kezdeni azzal, ha egy valós szoftverben a modellezett elemi átmenethez rendelt aktivitások módosítják a környezetet, vagyis nem vonhatók vissza?

- XSTS tranzíciók *atomiak*, Gamma \rightarrow XSTS leképezés az állapotgép teljes RTC lépését *egyetlen* XSTS tranzícióra képzí le.
- A szemantikailag atomi lépés elemi részeit csak akkor kezdhethetjük el „végrehajtani”, ha már megbizonyosodtunk róla, hogy *valóban végre tudjuk hajtani az egészet*. Ezzel elkerülhető, hogy bármit is „vissza kelljen vonni”.

2. kérd s

From the initial state s_0 , In is executed exactly once. Then, En and Tr are executed in alternation. In state s , the execution of a transition relation T (being either of the transition relations) means the execution of exactly one non-deterministically selected $t \in T$ transition. Transition t is enabled if $t(s) \neq \emptyset$. If a transition is not enabled, it can not be executed. If $\forall t \in T : t(s) = \emptyset$, transition relation T can not be executed in state s , and therefore s is a deadlock. In addition to the non-deterministic selection, transitions may be non-deterministic internally, therefore even in the case of a concrete state c , $t(c) = \{c'_1, \dots, c'_k\}$ may yield a set of successor concrete states. In other words, in the case of a general transition $t = (s, s')$, there is no restriction on the relation between $|s|$ and $|s'|$.

2. kérdés

[...] A kérdésről a 10. oldal alján ez szerepel: *"Note that assumptions may cause any composite operation to yield an empty set as the set of successor states. This allows us to use the choice operation as a guarded branching operator, ruling out branches where an assumption fails by yielding an empty set as the result of that branch. In this work, we make the following assumptions, which can be easily guaranteed by simple pre-processing."* – **mi az az "egyszerű előfeldolgozás", ami garantálja, hogy egy tetszőleges bonyolult elemi átmenet struktúra elkezdése valamilyen kiindulási állapotból biztosan nem okoz olyan változást a változók kiértékelésében, hogy a lánc egy későbbi "assumption"-ja ne teljesüljön?**

2. kérdés

[...] *"In this work, we make the following assumptions, which can be easily guaranteed by simple pre-processing."* – **mi az az "egyszerű előfeldolgozás", ami garantálja, hogy egy tetszőleges bonyolult elemi átmenet struktúra elkezdése valamilyen kiindulási állapotból biztosan nem okoz olyan változást a változók kiértékelésében, hogy a lánc egy későbbi "assumption"-ja ne teljesüljön?**

- Az idézett utolsó mondat már egy másik bekezdésbe tartozik, az említett „egyszerű előfeldolgozás” a a 11. oldal tetején lévő két *megszorítás* teljesülését garantálja.

2. kérdés

Note that assumptions may cause any composite operation to yield an empty set as the set of successor states. This allows us to use the *choice* operation as a guarded branching operator, ruling out branches where an assumption fails by yielding an empty set as the result of that branch.

In this work, we make the following assumptions, which can be easily guaranteed by simple pre-processing.

¹The default value of the type is used as an initializer unless explicitly specified by the modeler.

10

1. The operation of transitions and non-sequence composite actions must be composite actions. Thus, single basic operations will be treated as 1-long sequences.
2. We assume that there are no sequences directly inside sequences.

These restrictions help the clarity and consistency of local variable scopes without the loss of generality.

3. kérdés

A 11. példa kifejezésében az első sor a PC-nek 2-t ad értékül, majd az összes alatta levő sor "assumption"-jében $pc=1$ szerepel ÉS kapcsolatban valami mással: **ez helyes így?**

- *Nem*, nem helyes így, ez egy sajnálatos elírás.

Example 11 (Splitting parallel). For transition $t = ((x := 1, y := x) \parallel (x := 2, y := x))$ with a splittable parallel with two 2-long sequences, splitting will use 2 branch program counters pc_1, pc_2 , and result in 6 fragments:

$$\text{split}(t) = \text{split}(t, 0, 0) = \left\{ \begin{array}{l} ([pc = 0], pc_1 := 1, pc_2 := 1, pc := 2), \\ ([pc = 1 \wedge pc_1 = 1], x := 1, pc_1 := 2), \\ ([pc = 1 \wedge pc_1 = 2], y := x, pc_1 := 0), \\ ([pc = 1 \wedge pc_2 = 1], x := 2, pc_2 := 2), \\ ([pc = 1 \wedge pc_2 = 2], y := x, pc_2 := 0), \\ ([pc = 1 \wedge pc_1 = 0 \wedge pc_2 = 0], pc := 0) \end{array} \right\}$$

3. kérdés

4.3.5 Parallel

The splitting of a *parallel* of form $par = op_1 \parallel \dots \parallel op_n$ means splitting every operation of every branch into a separate fragment, as well as creating a fragment for *forking* and *joining* the branches. For every branch op_i , a separate *branch program counter* pc_i is introduced, in order to guarantee the execution order of operations from one branch: $V' = V \cup \{pc_1, \dots, pc_n\}$. The assumption on pc_i can be merged into the original pc assumption(s) at the beginning of the fragment with logical *and*.

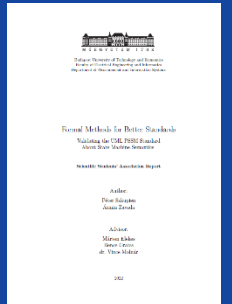
In order to keep the original control flow between *fork*, branches, and *join*, a new pc value ξ is needed. Formally, $split(par, x, y) = \{\text{forkfrag}(x, \xi, \bigcup_{i=1}^n pc_i), \bigcup_{i=1}^n \bigcup_{j=1}^{|op_i|} \text{parfrag}(\xi, pc_i, op_i, j), \text{joinfrag}(\xi, y, \bigcup_{i=1}^n pc_i)\}$.

The *fork* fragment $\text{forkfrag}(x, \xi, PC)$ checks $pc = x$, then assigns 1 to every branch program counter $pc_i \in PC$, and ξ to pc . Informally, the *fork* fragment enables the execution of the parallel branches. Formally, $\text{forkfrag}(x, \xi, PC) = ([pc = x], seq_{i=1}^{|PC|} PC_i := 1, pc := \xi)$, where $seq_{i=1}^n op_i$ means the sequence of op_1, \dots, op_n .

4. kérdés

6.3-ban felderített deadlock lehetőség az olvasónak azt sugallhatja, hogy az UML több évtizedes története során javasolt számtalan formalizálási próbálkozás után máig nem sikerült olyan szemantikát leírni a szabványban, amely legalább egy ilyen triviális modellre egyértelmű lenne. **Mikorra várható az UML-nek egy olyan verziója, amely mentes lesz ezektől a gyerekbetegségektől?**

- A látszólag triviális modellek szemantikájának bonyolultságát a szabvány által definiált *implicit komponensek* (pl. dispatcher) adják.
- A szabványok minőségének javítása érdekében célszerű lehet *formális módszerek* használata már a szabványosítási folyamat során is, amire javasoltunk egy módszert a TDK dolgozatunkban.



P. Szkupien, Á. Zavada:
Formal Methods for
Better Standards:
Validating the UML PSSM
Standard About State
Machine Semantics