Сети Петри и моделирование безопасных динамических ассоциаций

Frumin Daniil

Higher School of Economics, Dept of Software Engineering

February 26, 2013

Talk overview

Talk overview

Информация

Доклад основан на статье Fernando Rosa-Velardo (The Complutense University, Madrid)
“Petri nets with name creation for Transient Secure Association”

Мотивация

- Повсеместные вычисления (ubiquitous computing) изменили отношение к безопасности и способам авторизации
- Мы не можем больше надеяться на постоянно присутствующий единый сервер авторизации и аутентификации
- Одна из ослабленных форм аутентификации – временное безопасное соединение (transient secure association)
- Идея основана на “принципе (новорожденного) утенка” (Resurrecting Duckling Policy)

The Resurrecting Duckling

- The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks (Frank Stajano, Ross Anderson) – 7th International Workshop on Security Protocols
- Принцип утенка: первый движущийся и издающий звук объект, который утенок видит при рождении – мама.
- Таким же образом, “рожденные” устройство определит первое другое устройство, вступившее с ним в контакт, как родителя.
- “Resurrecting” означает то, что “утят” можно убивать, после чего они могут воскресать.

The Resurrecting Duckling

1. Принцип двойственности состояния (Two state principle): компонент может быть либо живым (alive, imprinted), либо мертвым (dead, imprintable);
2. Принцип импринтинга: импринтинг происходит, когда мама-утка посылает утенку специальный ключ;
3. Принцип смерти: устройство может умирать только когда его убивает мама-утка;
4. Принцип убийства: убийство своего утенка должно быть экономически не выгодно для утки.

Предварительные сведения

- Petri nets
- Petri nets w/ reset arcs
- 2CM
- Мультимножества, порядки

Предварительные сведения: мультимножества

- Пусть A – множество, тогда $A^{\oplus}$ будем называть множеством *конечных мультимножеств над* A
- Мультимножество – это отображение $m : A \to \mathbb{N}$, мы будем писать $a^n \in m$ если $m(a) = n$
- Поддержкой мультимножества (support) будем называть $supp(m) = \{a \in A \mid m(a) \neq 0\}$
- Обычным путем определены операции и отношения $+$, $-$, $\subseteq$ на мультимножествах
- Квазипорядок $\leq$ на A индуцирует квазипорядок $\leq^{\oplus}$ на $A^{\oplus}$: $\{a_1, a_2, \ldots, a_n\} \leq^{\oplus} \{b_1, b_2, \ldots, b_m\}$, если существует такая инъективная функция $h : \{0 \ldots n\} \to \{0 \ldots m\}$, что $a_i \leq b_{h(i)}$ для всех i

Предварительные сведения: деревья

- Пусть A – множество, тогда $\mathcal{T}(A)$ будем называть множеством *конечных (непустых) над* A
- Дерево над A – это пара (a, F), где $a \in A$ и $F \in \mathcal{T}(A)^{\oplus}$
- *Лесом* будем называть мультимножество деревьев, а *плоским лесом* – мультимножество листьев
- Высота дерева: $height((a, F)) = 1 + heigth(F)$, $heigth(F_1 + F_2) = max\{height(F_1), height(F_2)\}$, $height(\emptyset) = 0$

Предварительные сведения: деревья

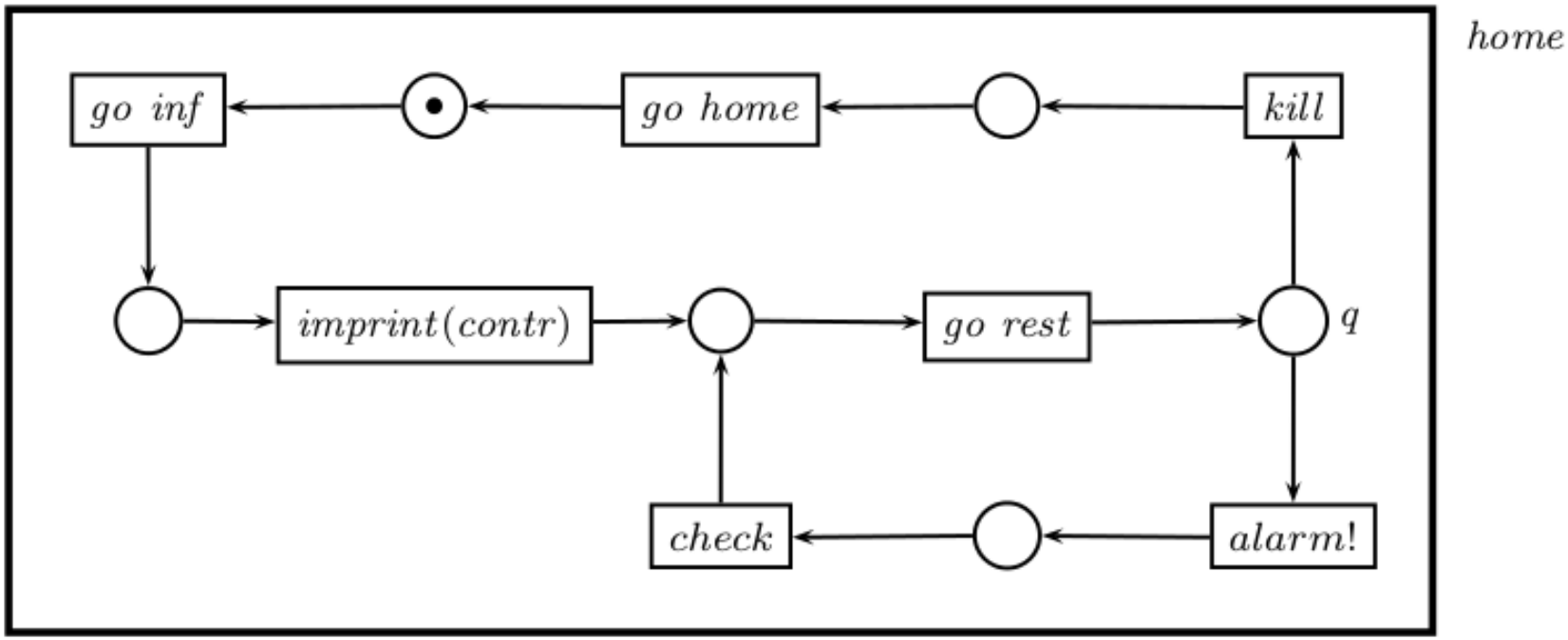
- Квазипорядок $\leq$ на A индуцирует квазипорядок $\leq_{\mathcal{T}(A)}$ на $\mathcal{T}(A)$: т.н. rooted order
- $(a, F) \leq_{\mathcal{T}(A)} (b, G) \iff a \leq b \wedge F \leq^{\oplus}_{\mathcal{T}(A)} G$

Talk overview

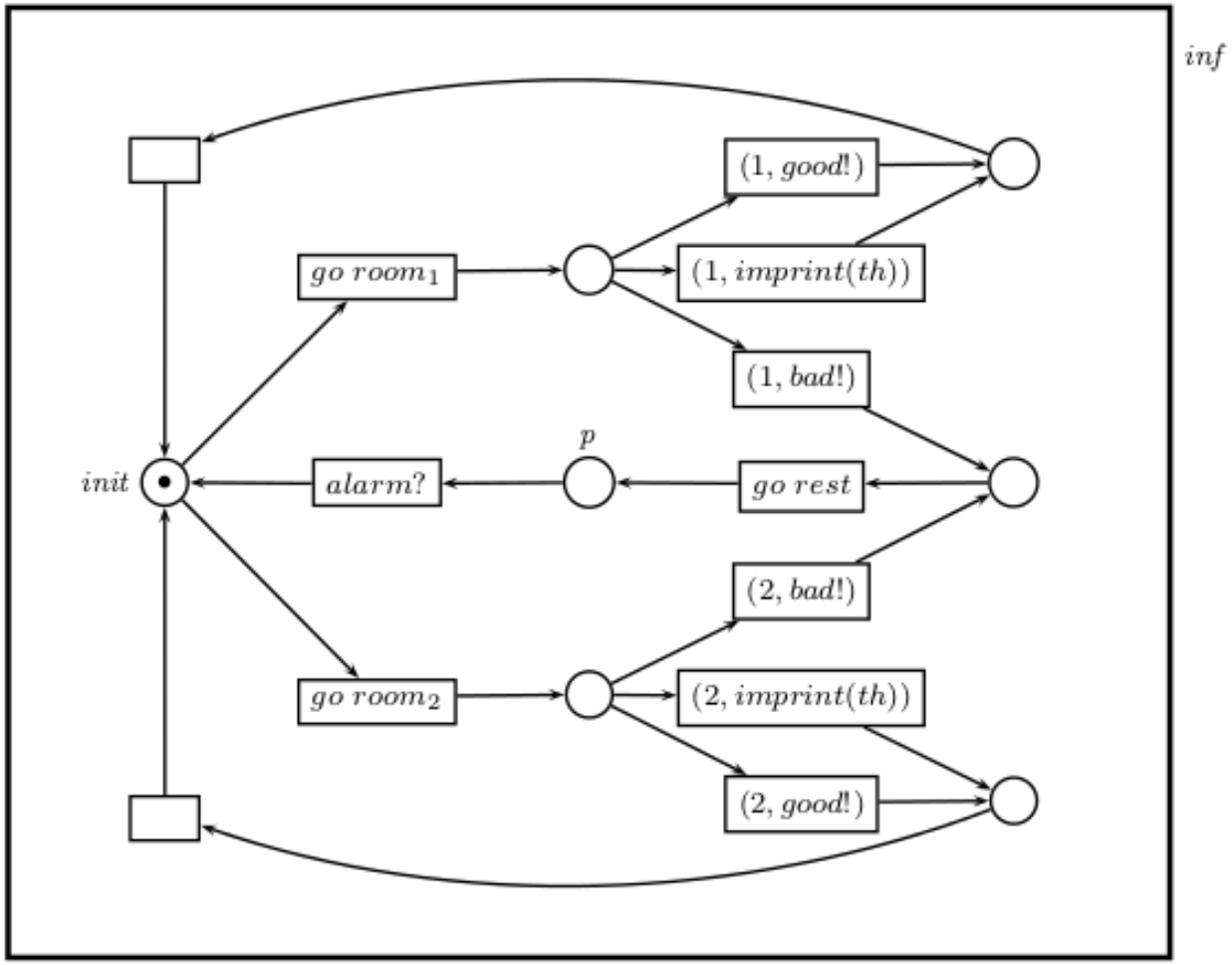
Описание

- TSA системы – это модель, основанная на сетях Петри с дополнительным функционалом для transient secure authentication;
- Есть пространство локаций, комнат - *Loc*,
- конечный набор компонент, каждая из которых является сетью и находится на какой-то локации и может передвигаться между локациями;
- Каждый компонент может заимпринтить другой компонент, отдавать приказы дочерним компонентам, убивать дочерние компоненты и сам становиться утенком.

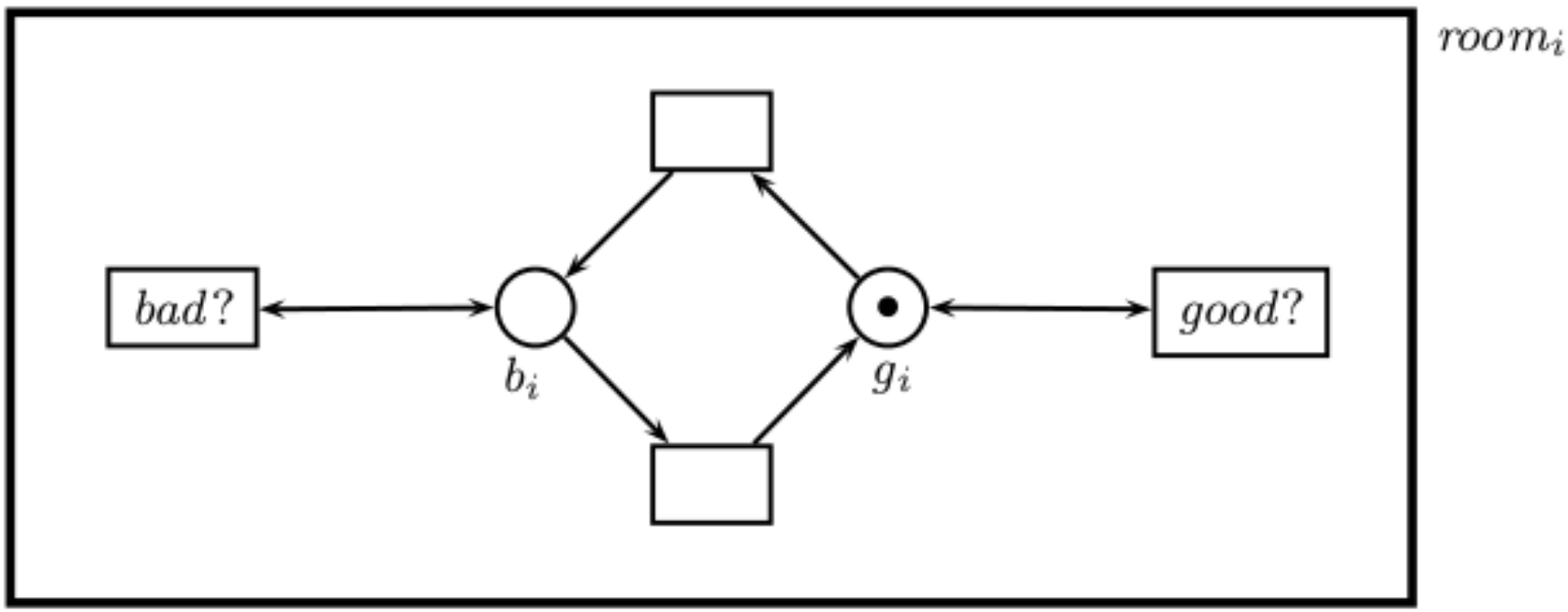
Пример



Пример


inf
(1, good!)
go room$_1$
(1, imprint(th))
(1, bad!)
p
init
alarm?
go rest
(2, bad!)
go room$_2$
(2, imprint(th))
(2, good!)

Пример



Формальное определение

Definition (TSA компонент)

TSA компонент – это кортеж $N = (P_N, T_N, F_N, \lambda_N, O_N, m_N)$, где (P_N, T_N, F_N) – это сеть Петри, $\lambda_N : T_N \to Lab$ – функция разметки переходов.

$Lab = \{\tau\} \cup \{go\ k \mid k \in Loc\} \cup \{s? \mid s \in S\} \cup$
$(\{1 \dots m\} \times (\{s! \mid s \in S\} \cup \{kill\} \cup \{imprint\ o \mid o \in \mathcal{O}\}))$

- $O_N \in \mathcal{O}$ – тип компонента;
- m_N – вместимость (capacity) компонента.

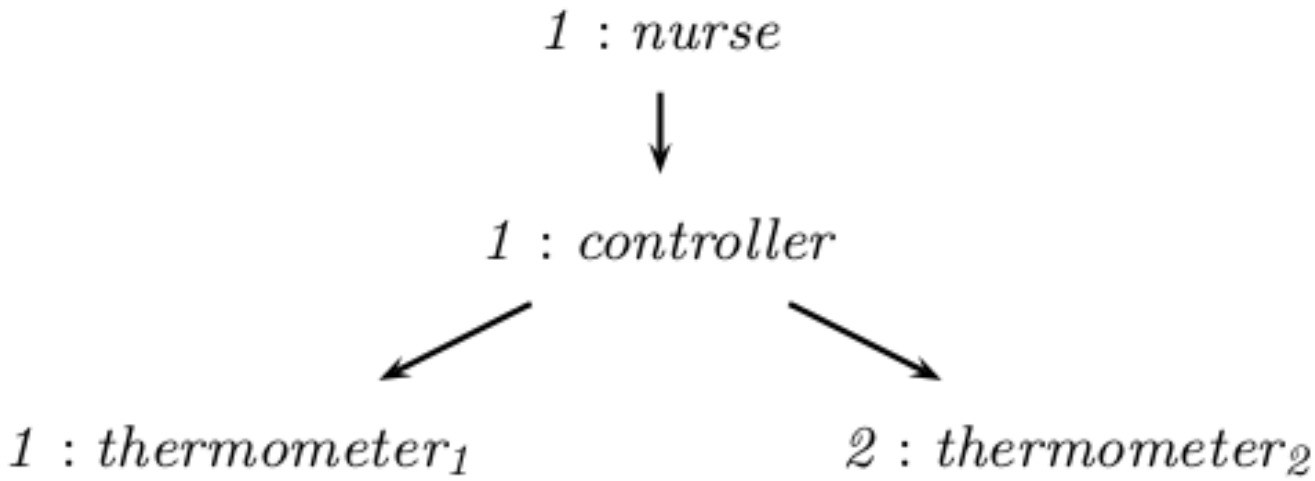
Формальное определение

Definition (TSA система)

TSA система – это (конечный) набор TSA компонент, вместе с Loc, $\mathcal{O}$ (который содержит специальный тип $\top$, который нельзя импринтить), набором команд S.

Компонент с типом $\top$ считаем всегда живым.

Пример: ассоциирование



Формальное определение

Definition (Ассоциирование)

Для TSA системы $\mathcal{N}$ ассоциирование – это лес над парами (k, N) где $k \in \mathbb{N}, N \in \mathcal{N}$.

Мы предполагаем, что ассоциирование R является допустимым, т.е. удовлетворяет условиям:

- Каждый компонент встречается в R лишь один раз;
- Компонент с типом $\top$ может быть только корнем в одном из деревьев в R;
- Корневой компонент имеет номер 1.

Формальное определение

Definition (Конфигурация)

Конфигурация TSA системы $\mathcal{N}$ это тройка $C = (M, loc, R)$, где $M \in P_{\mathcal{N}}^{\oplus}$ – разметка всей системы, $loc : \mathcal{N} \to Loc$, R – ассоциирование системы.

Графически, локации отображаются плоскими коробками, в которых находятся (loc) размеченные (M) компоненты.

Формальное определение: поведение

Далее и на других слайдах: $\mathcal{N}$ – TSA система с конфигурацией $C = (M, loc, R)$

Definition (Автономный шаг)

Пусть N – живой компонент в R, и $t \in T_N$.

- Если $pre(t) \subseteq M$ и $\lambda_N(t) = \tau$, то t может сработать и получить конфигурацию $C' = ((M - pre(t)) + post(t), loc, R)$
- Если $pre(t) \subseteq M$ и $\lambda_N(t) = go\ k$, то t может сработать и получить конфигурацию $C' = ((M - pre(t)) + post(t), loc[N := k], R)$

Формальное определение: поведение

Definition (Шаг с импринтингом)

Пусть N – живой компонент, а N' – мертвый компонент, находящийся в той же локации, что и N. Пусть $t \in T_N$ и $\lambda_N(t) = (i, imprint(O_{N'}))$. Тогда

- t может сработать в N если $pre(t) \subseteq M$ и N не имеет i-ого потомка в R
- и мы получим конфигурацию $C' = ((M - pre(t)) + post(t), loc, R')$, где R' – расширение R, где (i, N') является потомком N.

Формальное определение: поведение

Definition (Шаг с убийством)

Пусть N – живой компонент, а N' – живой компонент, находящийся в той же локации, что и N. Пусть $t \in T_N$ и $\lambda_N(t) = (i, kill)$. Тогда

- t может сработать в N если $pre(t) \subseteq M$ и N' является i-ым потомком N в R
- и мы получим конфигурацию $C' = ((M - pre(t)) + post(t), loc, R')$, где R' – это ассоциирование, полученное удалением из R поддерева с (i, N') в качестве корня.

Формальное определение: поведение

Definition (Синхронизированный переход)

Пусть N – живой компонент, а N' – живой компонент, находящийся в той же локации, что и N. Пусть $t \in T_N$ и $\lambda_N(t) = (i, s!)$. Пусть $t \in T_{N'}$ и $\lambda_{N'}(t) = s?$. Тогда

- t и t' могут сработать одновременно если $pre(t) + pre(t') \subseteq M$ и N' является i-ым потомком N в R
- и мы получим конфигурацию $C' = (M - (pre(t) + pre(t')) + post(t) + post(t'), loc, R)$.

TSA системы и P/T-сети

Theorem

Каждая TSA система изоморфна (с точки зрения порядка) какой-то P/T-сети.

Вкупе с тем фактом, что P/T-сеть является частным случаем TSA системы (один компонент, одна локации, нулевая вместимость), мы получаем, что TSA системы эквивалентны P/T-сетям, что означает разрешимость многих задач для TSA систем.

Talk overview

TSA с инициализацией компонент

Как мы можем заметить, из определения перехода с убийством, следует что убитый компонент остается все в той маркировке, в которой его убили, до того самого момента, когда его снова оживят.
Иногда нам требуется, что бы убитые устройства возвращались в изначальные состояния – reset'ились. Это, например, бывает нужно, когда требуется сохранить конфиденциальность каких-либо данных, которые могут быть отражены в состоянии компонента.

TSA с инициализацией компонет

В этих целях мы расширяем определение компонента:

$$N = (P_N, T_N, F_N, \lambda_N, O_N, m_N, init_N)$$

Теперь компонент содержит инициализирующую маркировку $init_N \in P_N^{\oplus}$.

TSA с инициализацией компонент: поведение

Definition (Шаг с убийством)

Пусть N – живой компонент, а N' – живой компонент, находящийся в той же локации, что и N. Пусть $t \in T_N$ и $\lambda_N(t) = (i, kill)$. Пусть так же $\mathcal{N}'$ – множество детей компонента N' в R (включая сам N'). Тогда

- t может сработать в N если $pre(t) \subseteq M$ и N' является i-ым потомком N в R
- и мы получим конфигурацию $C' = (M', loc, R')$, где R' – это ассоциирование, полученное удалением из R поддерева с (i, N') в качестве корня
- $M' = M - (pre(t) + M|_{\mathcal{N}'}) + post(t) + \sum_{c \in \mathcal{N}'} init(c)$, где $M|_{\mathcal{N}'}$ – ограничение разметки M на $\mathcal{N}'$

TSA системы и P/T-сети

Theorem

Каждая TSA система с инициализацией изоморфна (с точки зрения порядка) какой-то P/T-сети с reset arcs.

Theorem

Для каждой P/T-сети с reset arcs можно построить TSA систему с инициализацией, которая будет симулировать ее.

Таким образом мы получаем, что P/T-сети эквивалентны TSA системам с инициализацией.

TSA системы и P/T-сети

Theorem

Каждая TSA система с инициализацией изоморфна (с точки зрения порядка) какой-то P/T-сети с reset arcs.

Theorem

Для каждой P/T-сети с reset arcs можно построить TSA систему с инициализацией, которая будет симулировать ее.

Таким образом мы получаем, что P/T-сети эквивалентны TSA системам с инициализацией. Проблемы достижимости и ограниченности не разрешимы для TSA систем с инициализацией.

Talk overview

uTSA

До текущего момента, мы исходили из предположения, что в нашей среде находится фиксированное количество компонент, каждый из которых имеет определенную вместимость и не может импринтить больше чем какое-то количество компонент.

uTSA

До текущего момента, мы исходили из предположения, что в нашей среде находится фиксированное количество компонент, каждый из которых имеет определенную вместимость и не может импринтить больше чем какое-то количество компонент. В данной секции мы рассмотрим модель, в которой есть один тип компоненты, но который может создавать (и импринтить) неограниченное число компонент такого же типа. Например, это программа, которая может запускать сама себя в новом потоке.

uTSA

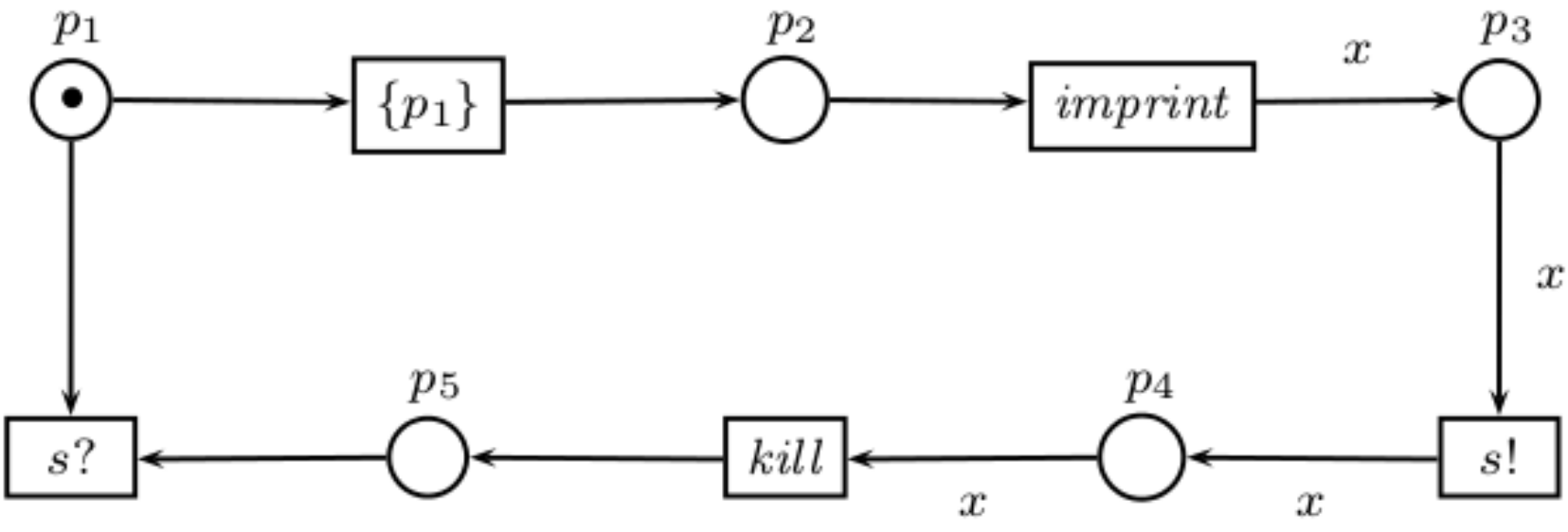
С технической точки зрения это значит что у нас есть какая-то сеть, которая может создавать копии себя, с определенной маркировкой. Такие созданный компоненты изначально являются мертвыми, но могут быть оживленны.
В результате, дерево ассоциациирования получается (потенциально) неограниченным.

uTSA

С технической точки зрения это значит что у нас есть какая-то сеть, которая может создавать копии себя, с определенной маркировкой. Такие созданный компоненты изначально являются мертвыми, но могут быть оживленны.
В результате, дерево ассоциациирования получается (потенциально) неограниченным.
Таким образом, для простоты мы вводим следующие ограничения:

- $\mathcal{O}$ является синглтоном (и вместо $imprint(o)$ у нас будет просто команда $imprint$;
- Loc является синглтоном (и у нас не будет команды go).

uTSA: пример



uTSA: формальное определение

Definition (uTSA)

Неограниченная TSA система (uTSA system) – это кортеж $N = (P, T, F, W, \lambda)$, где (P, T, F) – это сеть Петри, где

- В качестве фишек (tokens) используются элементы множества $Id \cup \{\bullet\}$, где Id – бесконечное множество идентификаторов;
- $\lambda \to Lab$ – функция разметки переходов и $Lab = S? \cup S! \cup \{imprint, kill, \tau\} \cup P^{\oplus}$
- W – функция $F \to \{x, \epsilon\}$ сопоставляющая каждой дуге из F выражение x или пустое выражение.

uTSA: формальное определение

Definition (uTSA компонент)

Компонент uTSA – это uTSA система N с разметкой.
Именованный uTSA компонент – это пара (i, C), где $i \in Id$, а C – uTSA компонент.

Мертвый компонент M будем обозначать M^d.
Для простоты, мы будем считать, что в системе есть живой компонент *top*.

uTSA: активные переходы и шаги

Для того, что бы определить семантику переходов в системе, мы воспользуемся функциями оценивания. Для каждого $a \in Id$ мы определяем σ_a как функцию $\sigma(x) = a$, $\sigma(\epsilon) = \bullet$, расширенную на $\{x, \epsilon\}^{\oplus}$.

Definition (Переходы в uTSA системе)

Пусть M – компонент uTSA системы N, $t \in T$ – переход и $a \in Id$. Переход t считается активным (может сработать) при обозначении a, если для каждого $p \in P$, $\sigma_a(W(p,t)) \subseteq M(p)$. В таком случае переход может сработать и мы получим маркировку $M'(p) = (M(p) - \sigma_a(W(p,t)) + \sigma_a(W(t,p))$.

uTSA: семантика

В случае срабатывания какого-либо перехода t с обозначением a мы будем писать $M \xrightarrow{\alpha} M'$, где M' – полученная маркировка, а α определяется следующим образом:

- $\alpha = \tau$, если $\lambda(t) = \tau$
- $\alpha = s!(a)$, если $\lambda(t) = s!$
- $\alpha = s?$, если $\lambda(t) = s?$
- $\alpha = b : \overline{M}^d$, если $\lambda(t) = \overline{M}$, т.е. $\overline{M}$ – компонент с разметкой $\overline{M}$, а b – новый идентификатор, не использующийся в M
- $\alpha = imp(a : \overline{M})$, если $\lambda(t) = imprint$, а $\overline{M}$ – компонент с маркировкой $\overline{M}$
- $\alpha = kill(a)$, если $\lambda(t) = kill$

uTSA: семантика

Definition (Конфигурация uTSA системы)

Конфигурация C uTSA системы N – это лес над множеством именованных компонент N, с условием, что все имена в C различны.

Под верхней конфигурацией будем подразумевать специальную конфигурацию $(top, C_1 + C_2)$, где C_1 – живые компоненты, C_2 – мертвые.

uTSA: семантика, обозначения

Мы будем использовать $Id(C)$ как обозначение множества имен компонент встречающихся в C.

- $Id((a : M, C)) = \{a\} \cup Id(C)$
- $Id(C_1 + C_2) = Id(C_1) \cup Id(C_2)$
- $Id(\emptyset) = \emptyset$

Под $flat(C)$ мы будем понимать мультимножество компонент, встречающихся в C.

- $flat((a : M, C)) = \{a : M_\bullet^d\} \cup flat(C)$
- $flat(C_1 + C_2) = flat(C_1) \cup flat(C_2)$
- $flat(\emptyset) = \emptyset$

где $M_\bullet$ получается из M заменой всех токенов на $\bullet$.

uTSA: семантика, обозначения

Под $M|_{\neq a}$ будем подразумевать разметку M с убранными токенами a. Т.е. для всех $p \in P$

- $M|_{\neq a}(p)(a) = 0$
- $M|_{\neq a}(p)(b) = M(p)(b)$ для $a \neq b$

uTSA: семантика

Переходы конфигураций (transitions of configurations) имеют следующий вид:

- $C \xrightarrow{\tau} C'$ – C переходит в C' без каких либо изменений общего состояния;
- $C \xrightarrow{imp(a:M)} C'$ – C переходит в C' с помощью импринтинга мертвого компонента M^d;
- $C \xrightarrow{\gamma} C'$, где γ - мультимножество мертвых компонет – C переходит в C' поднимая в верхнюю конфигурацию компоненты γ.

uTSA: операционная семантика

$$\frac{M \xrightarrow{\tau} M'}{(a{:}M, C) \xrightarrow{\tau} (a{:}M', C)}\,(\tau) \qquad \frac{M \xrightarrow{b:\overline{M}} M' \quad b \neq a \quad b \notin Id(C)}{(a{:}M, C) \xrightarrow{b:\overline{M}} (a{:}M', C)}\,(\mathrm{Rep})$$

$$\frac{M_1 \xrightarrow{s!(b)} M_1' \qquad M_2 \xrightarrow{s?} M_2'}{(a{:}M_1, (b{:}M_2, C_1) + C_2) \xrightarrow{\tau} (a{:}M_1', (b{:}M_2', C_1) + C_2)}\,(\mathrm{Sync})$$

$$\frac{M \xrightarrow{imp(b:\overline{M})} M'}{(a{:}M, C) \xrightarrow{imp(b:\overline{M})} (a{:}M', C + b{:}\overline{M})}\,(\mathrm{Imp})$$

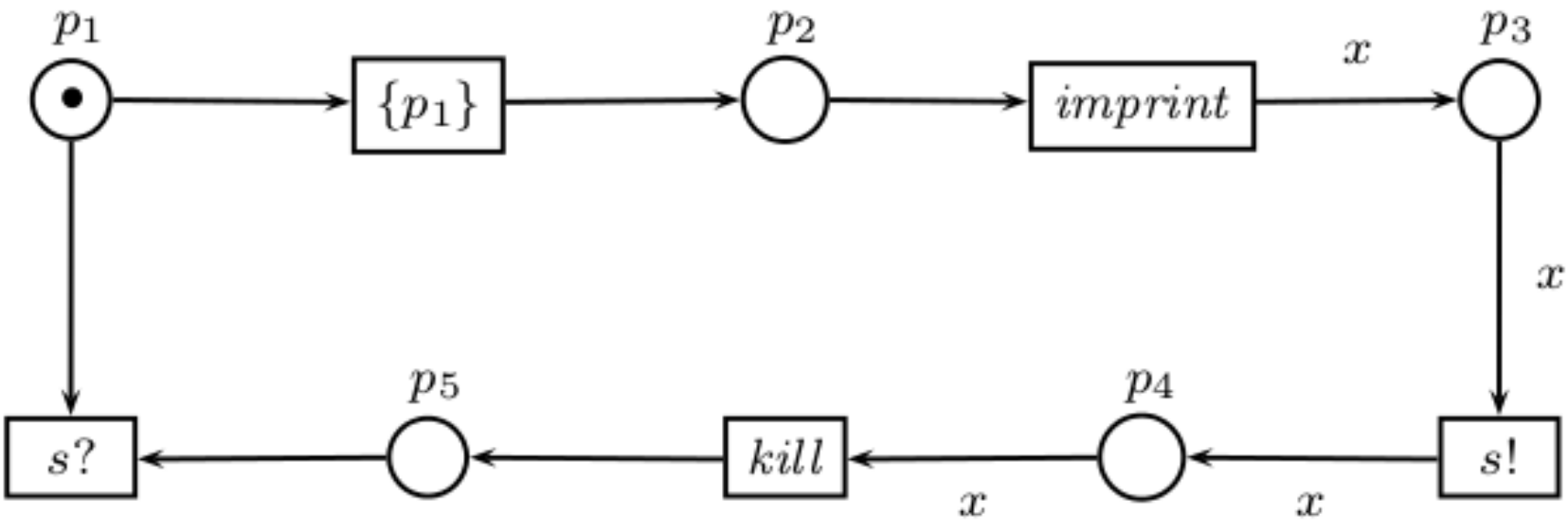
$$\frac{M \xrightarrow{kill(b)} M'}{(a{:}M, (b{:}\overline{M}, C_1) + C_2) \xrightarrow{flat((b:\overline{M}, C_1))} (a{:}M'|_{\neq b}, C_2)}\,(\mathrm{Kill})$$

uTSA: операционная семантика

$$\frac{C_1 \xrightarrow{\gamma} C_2 \quad Id(\gamma) \cap Id(C) = \emptyset}{C_1 + C \xrightarrow{\gamma} C_2 + C} \text{ (Hor)} \qquad \frac{C_1 \xrightarrow{\gamma} C_2 \quad a \notin Id(\gamma)}{(a{:}M, C_1) \xrightarrow{\gamma} (a{:}M, C_2)} \text{ (Ver)}$$

$$\frac{C_1 \xrightarrow{\mathcal{M}} C_2}{(top, C_1) \xrightarrow{\tau} (top, C_2 + \mathcal{M})} \text{ (Top1)} \qquad \frac{C_1 \xrightarrow{imp(b:M)} C_2}{(top, C_1 + b{:}M^d) \xrightarrow{\tau} (top, C_2)} \text{ (Top2)}$$

uTSA: пример



uTSA: пример

$$\frac{p_1 \xrightarrow{b:p_1^d} p_2}{(a:p_1) \xrightarrow{b:p_1^d} (a:p_2)}\ (Rep)$$

$$\frac{(a:p_1) \xrightarrow{b:p_1^d} (a:p_2)}{(top,(a:p_1)) \xrightarrow{\tau} (top,(a:p_2)+(b:p_1^d))}\ (Top1)$$

uTSA: пример

$$\dfrac{\dfrac{p_2 \xrightarrow{imp(b:p_1)} p_3[b]}{(a:p_2) \xrightarrow{imp(b:p_1)} (a:p_3[b],(b:p_1))}\,(Imp)}{(top,(a:p_2)+(b:p_1^d)) \xrightarrow{\tau} (top,(a:p_3[b],(b:p_1)))}\,(Top2)$$

uTSA: пример

$$\dfrac{\dfrac{p_3[b] \xrightarrow{s!(b)} p_4[b] \quad p_1 \xrightarrow{s?} 0}{(a : p_3[b], (b : p_1)) \xrightarrow{\tau} (a : p_4[b], (b : 0))}\,(Sync)}{(top, (a : p_3[b], (b : p_1))) \xrightarrow{\tau} (top, (a : p_4[b], (b : 0)))}\,(Ver)$$

uTSA: пример

$$\dfrac{\dfrac{p_4[b] \xrightarrow{kill(b)} p_5}{(a : p_4[b], (b : 0)) \xrightarrow{b:0^d} p_5}\,(Kill)}{(top, (a : p_4[b], (b : 0))) \xrightarrow{\tau} (top, (a : p_4[b]) + (b : 0^d)}\,(Top1)$$

uTSA и машины Тьюринга

Theorem (uTSA и 2CM)

Для любой 2CM найдется uTSA система, симулирующая её.

Идея: моделирование двух счетчиков с помощью двух ветвей дерева отношения потомок-родитель.

uTSA и машины Тьюринга

Theorem (uTSA и 2CM)

Для любой 2CM найдется uTSA система, симулирующая её.

Идея: моделирование двух счетчиков с помощью двух ветвей дерева отношения потомок-родитель. Следствие: uTSA системы эквивалентны по мощности машинам Тьюринга.

uTSA with bounded height

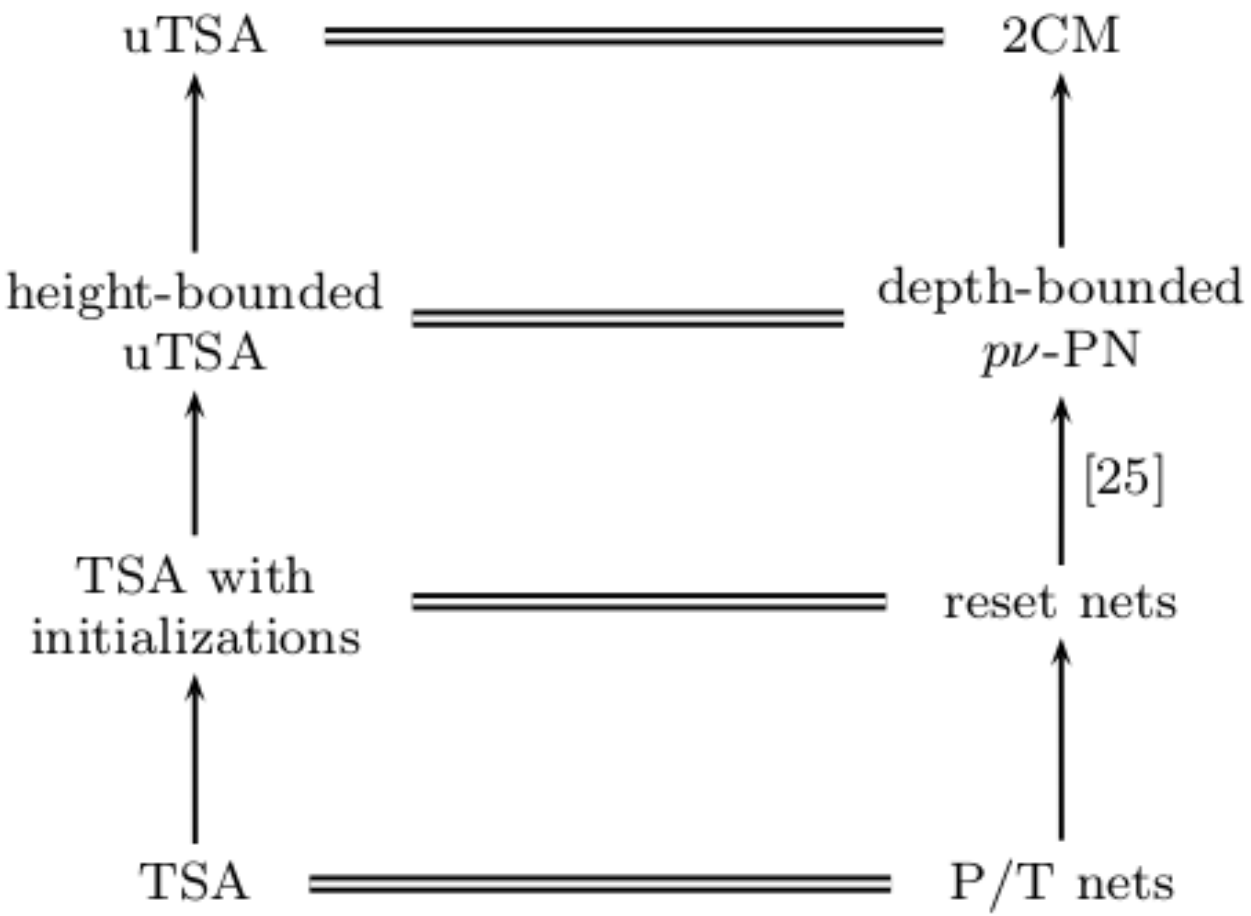
Идея: рассматривать uTSA системы с ограниченной глубиной.

uTSA with bounded height

Идея: рассматривать uTSA системы с ограниченной глубиной.
Height-bounded uTSA эквивалентны depth-bounded $\rho\nu-$PN

Talk overview

Conclusions



Спасибо за внимание!