

Верификация программ на моделях

Лекция №9

Выразительная мощность LTL.

Корректная абстракция графов программ.

Отношение моделирования (симуляции).

Константин Савенков (лектор)

План лекции

- Выразительная мощность LTL:
 - сравнение с автоматами Бюхи,
 - сравнение с CTL*, CTL,
 - Общая картина.
- Корректная абстракция графов программ:
 - Отношение слабого моделирования,
 - Сильное моделирование.

Выразительная мощность LTL по сравнению с конструкциями never

- Автомат Бюхи описывает ω -регулярный язык:
 - A^ω , где A – регулярный язык,
 - AB , где A – регулярный, а B – ω -регулярный язык,
 - $A \cup B$, где A и B – ω -регулярные языки;
- при помощи never можно описать любой ω -регулярный автомат над словами.

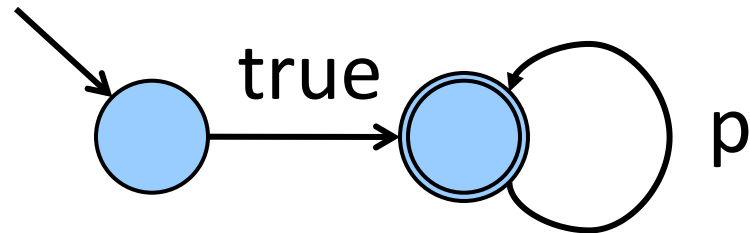
Выразительная мощность LTL по сравнению с конструкциями never

- LTL описывает подмножество этого языка:
 - всё, выразимое в LTL, может быть описано при помощи never,
 - при помощи never можно описать свойства, невыразимые на LTL.
- **Теорема:** Добавление одного квантора существования над одним пропозициональным символом расширяет выразительные способности LTL до всех ω -регулярных автоматов над словами.

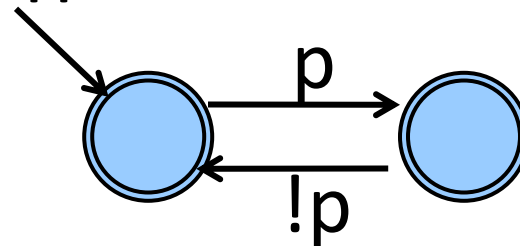
Пример свойства, не выражимого на LTL

- (p) **может** быть истинным после выполнения системой чётного числа шагов, но **никогда не истинно** после **нечётного**.

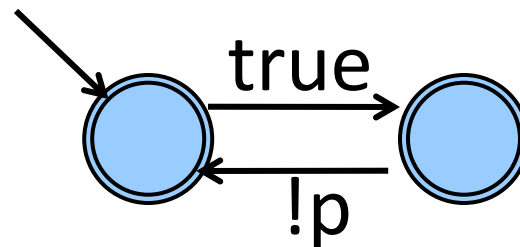
- $[]X(p)$ не подходит



- $p \ \&\& \ [](p \rightarrow X!p) \ \&\& \ [](!p \rightarrow Xp)$ – также не подходит (здесь p **всегда** истинно после чётных шагов)



- $\exists t. t \ \&\& \ [](t \rightarrow X!t) \ \&\& \ [](!t \rightarrow Xt) \ \&\& \ [](!t \rightarrow !p)$ – то, что надо:



Сравнение LTL с другими логиками

- LTL-формула описывает свойство, которое должно выполняться на **всех** вычислениях, начинающихся из исходного состояния системы

Операторы:

!	логическое отрицание
&&	логическое И
	логическое ИЛИ
X	в следующем состоянии
U	сильный until
U	слабый until
<>	рано или поздно
[]	всегда

Серым отмечены операторы, которые можно вывести из других:

$\varphi_1 \parallel \varphi_2$	==	$!(\neg \varphi_1 \&\& \neg \varphi_2)$
$\langle \rangle \varphi$	==	$\text{true} \cup \varphi$
$[] \varphi$	==	$\neg \langle \rangle \neg \varphi$
$\varphi_1 \cup \varphi_2$	==	$[] \varphi_1 \parallel ([\varphi_1 \cup \varphi_2])$

грамматика:

пропозициональные формулы:

p
!f
(f)
f && f
f || f

темпоральные формулы:

f
! φ
(φ)
 $\varphi \&\& \varphi$
 $\varphi \parallel \varphi$
X φ
 $\varphi \cup \varphi$
 $\langle \rangle \varphi$
 $[] \varphi$

p – некоторый пропозициональный символ

f – некоторая пропозициональная формула

φ – некоторая темпоральная формула

Логика STL*

- Логика ветвящегося времени:
 - использует кванторы $\forall$ и $\exists$,
 - использует F вместо $\langle \rangle$ и G вместо $[]$.

Операторы:

!	логическое отрицание
&&	логическое И
	логическое ИЛИ
E	существует путь
A	для всех путей
X	в следующем состоянии
U	until (сильный)
F	рано или поздно
G	всегда

Серым отмечены операторы, которые можно вывести из других:

$\varphi_1 \varphi_2$	==	$!(\neg \varphi_1 \&\& \neg \varphi_2)$
$A \varphi$	==	$!E \neg \varphi$
$F \varphi$	==	$\text{true} U \varphi$
$G \varphi$	==	$!F \neg \varphi$

формулы состояния: p

!f
(f)
f && f
f || f

A φ

E φ

формулы пути:

f
! φ
(φ)
 $\varphi \&\& \varphi$
 $\varphi || \varphi$
X φ
 $\varphi U \varphi$
F φ
G φ

p – некоторый пропозициональный символ

f – некоторая формула состояния

φ – некоторая формула пути

Логика CTL

- Логика CTL – фрагмент логики CTL*, в котором под управлением квантора пути (E или A) может находиться не более одного оператора X или U.

Корректная CTL формула:

p
 $! \varphi$
 $\varphi \ \&\& \ \varphi$
 $\varphi \ || \ \varphi$
 $E \ X \ \varphi$
 $E \ (\varphi \ U \ \varphi)$
 $A \ (\varphi \ U \ \varphi)$

p – некоторый пропозициональный символ

f – некоторая формула состояния

φ – некоторая формула пути

Можно вывести:

EF f	==	E(true U f)
AF f	==	A(true U f)
EG f	==	!AF !f
AG f	==	!EF !f
AX f	==	!EX !f

Пример

В LTL $\langle \rangle p$ означает:

$A\langle \rangle p$ **для всех** вычислений, начинающихся
в исходном состоянии s_0 , выполняется $\langle \rangle p$

В CTL можно выразить:

$EF(p)$ **существует** вычисление, для которого
выполняется $\langle \rangle p$

$AF(p)$ **для всех** вычислений выполняется $\langle \rangle p$

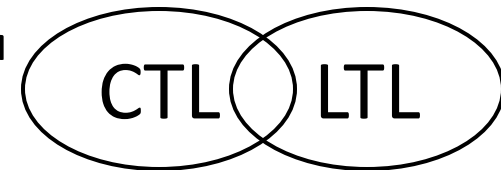
$AG(p)$ **для всех** вычислений p – инвариант

$EG(p)$ **существует** вычисление, для которого
выполняется p – инвариант

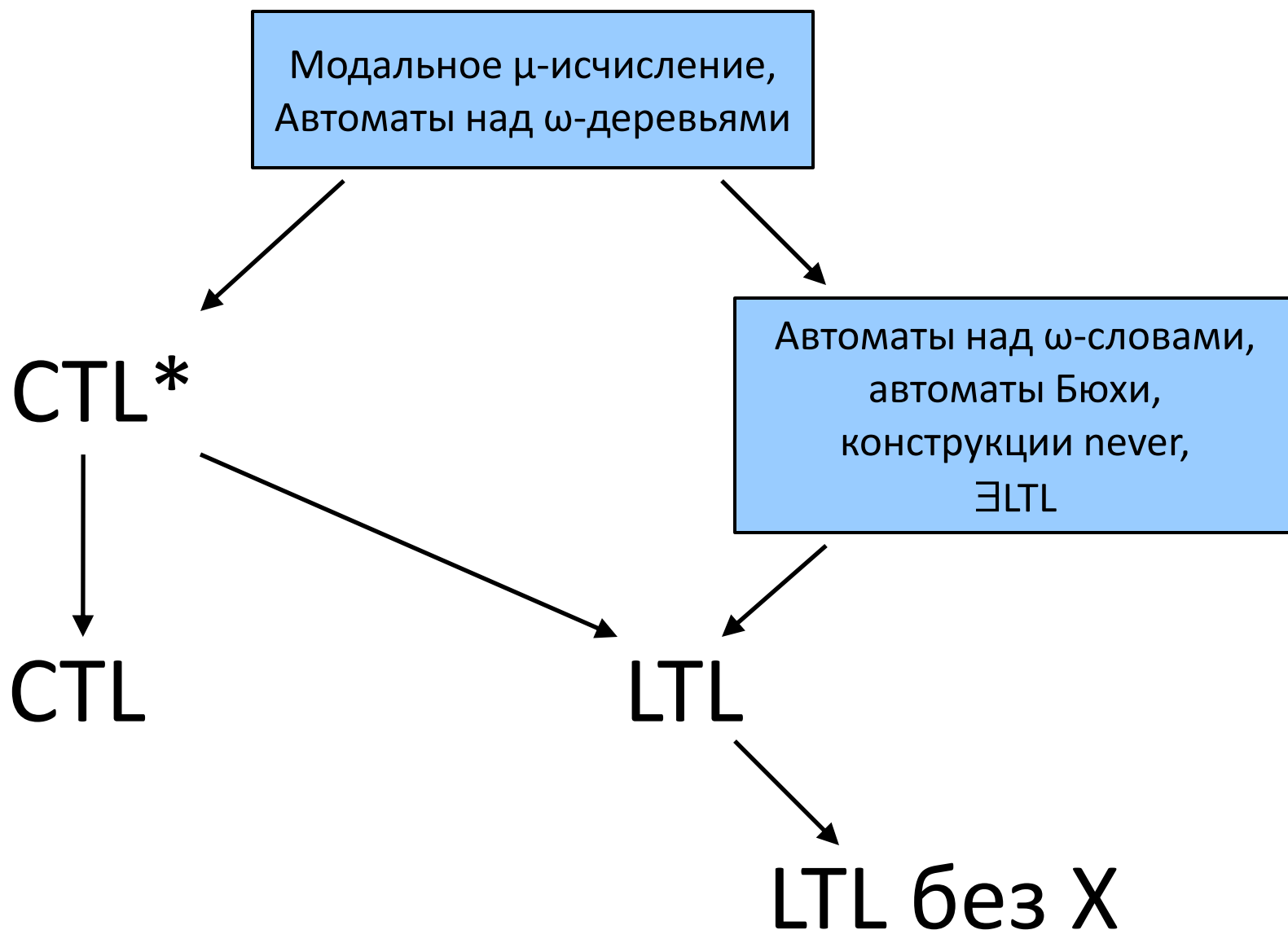
итд.

Выразительные возможности CTL* и CTL

- CTL* и CTL описывают подмножества ω -регулярных автоматов над *деревьями*
 - автоматы над деревьями более выразительны, чем автоматы над словами (CTL-формула выполнима на дереве трасс, а не на одной трассе);
 - CTL и LTL являются подмножествами CTL*;
 - CTL и LTL не сравнимы по выразительной мощности (пересекаются, но не включают);
 - на LTL можно описать свойства, не выразимые на CTL:
 - CTL не позволяет описать свойства вида $[\]\langle \rangle(p)$,
 - при помощи $[\]\langle \rangle(p)$ в LTL задаются ограничения справедливости;
 - на CTL можно описать свойства, не выразимые на LTL:
 - на LTL нельзя описать свойства вида $\text{AGEF}(p)$,
 - $\text{AGEF}(p)$ используется для описания свойства reset: из любого состояния система может перейти в нормальное.

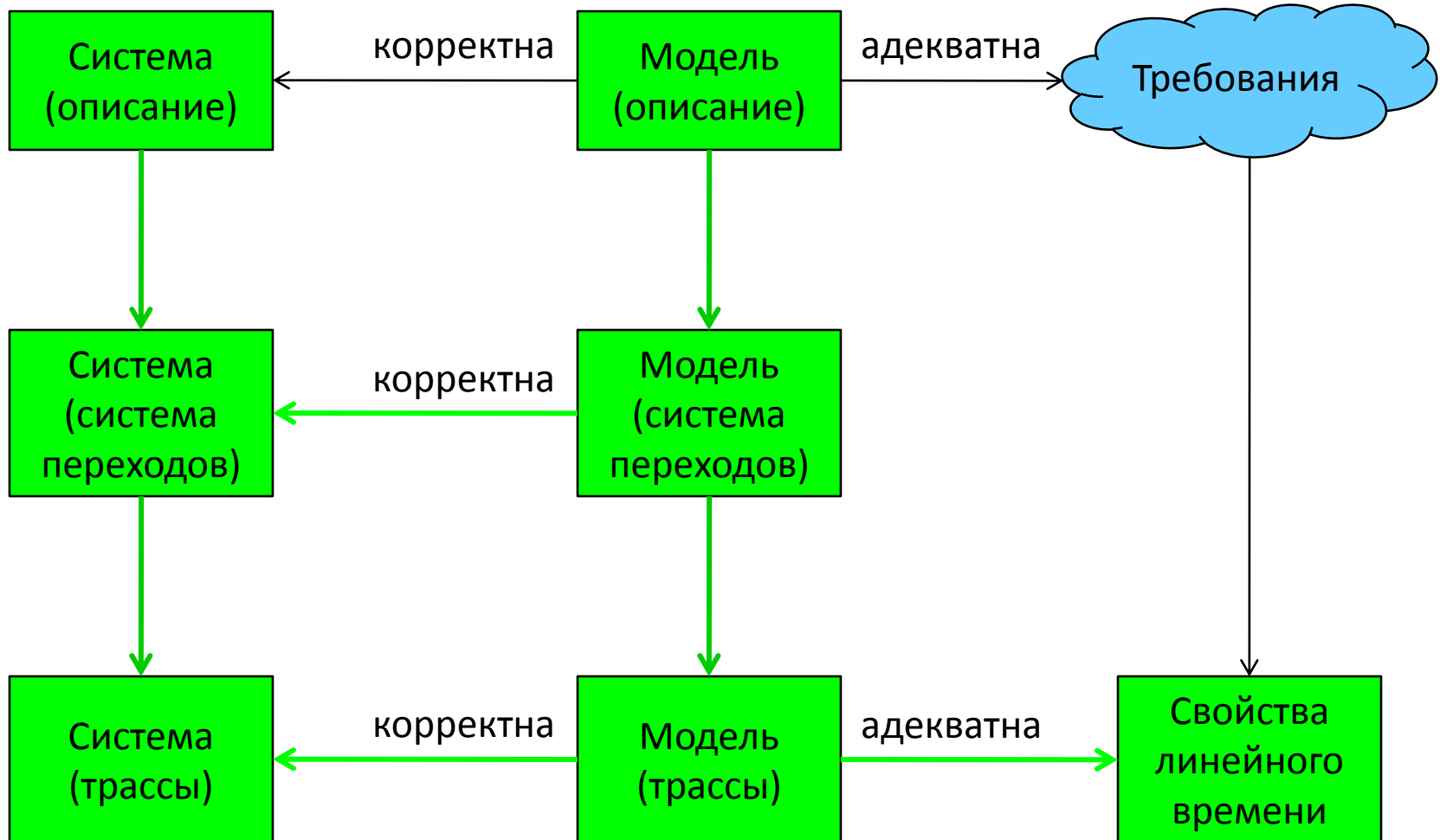


Выразительная мощность

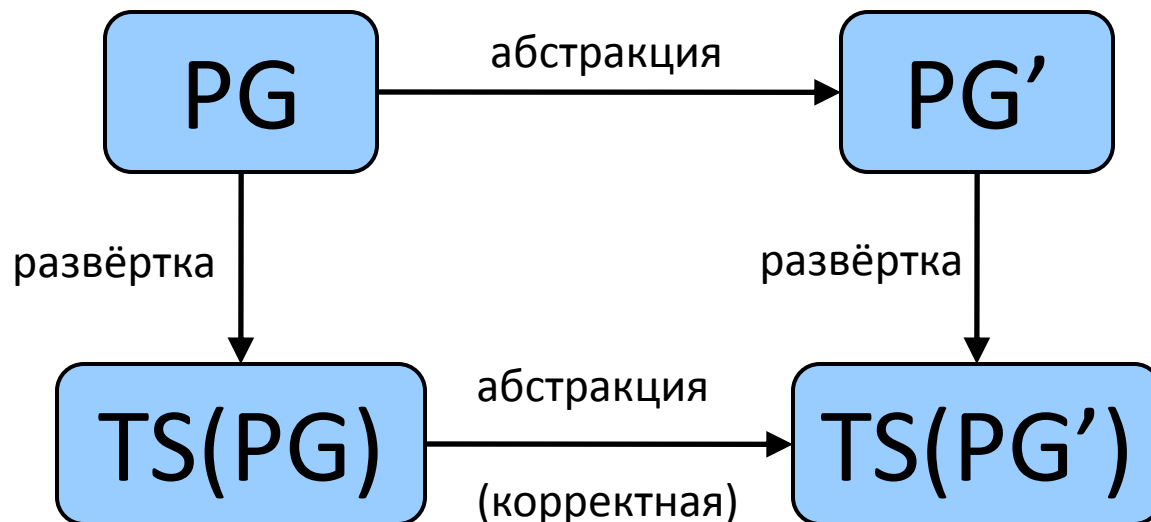


Корректность абстракции графов программ

Схема понятий



Корректность моделирования графов программ



- **Необходимое и достаточное условие:**
 - Программа PG' корректно моделирует программу PG тогда и только тогда, когда система переходов $TS(PG')$ корректно моделирует систему переходов $TS(PG)$.

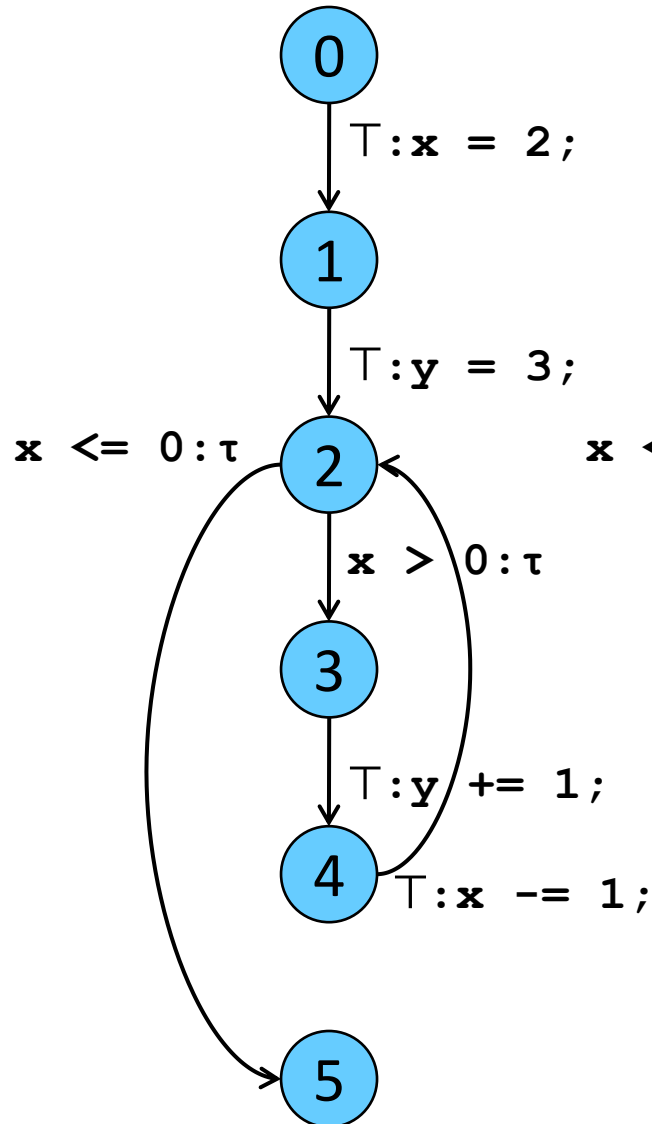
Абстракция графов программ

Пример корректной абстракции

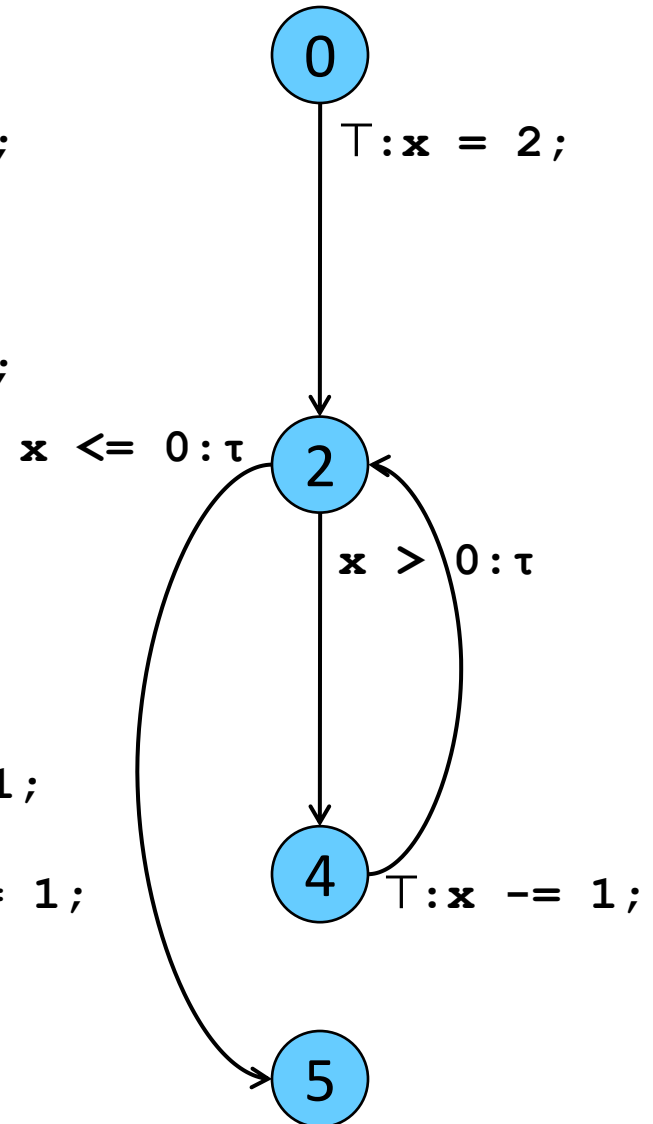
Программа:

```
int x, y;  
0: x = 2;  
1: y = 3;  
2: while (x > 0)  
  {  
3:   y += 1;  
4:   x -= 1;  
  }  
5:
```

PG:

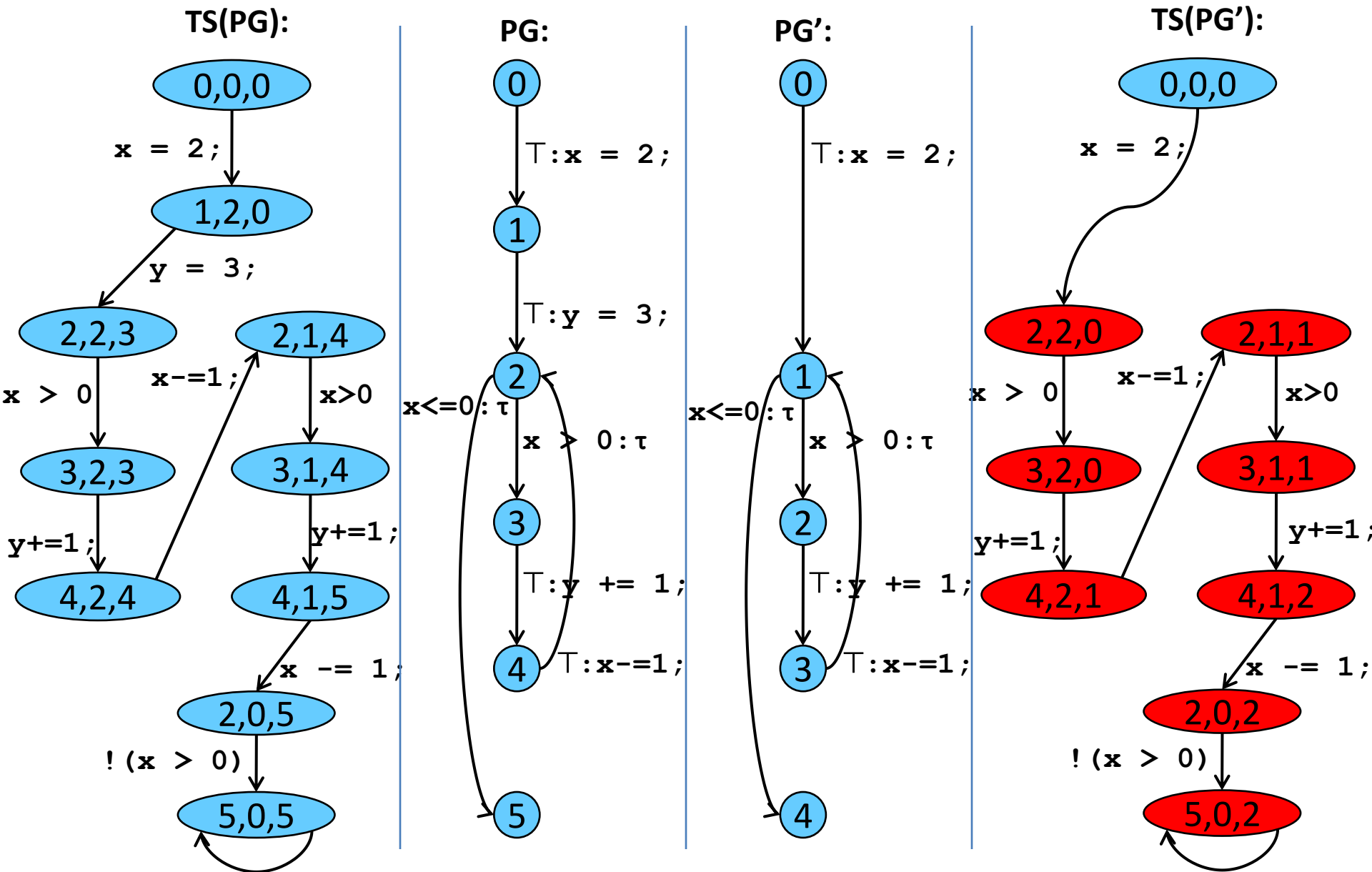


PG':



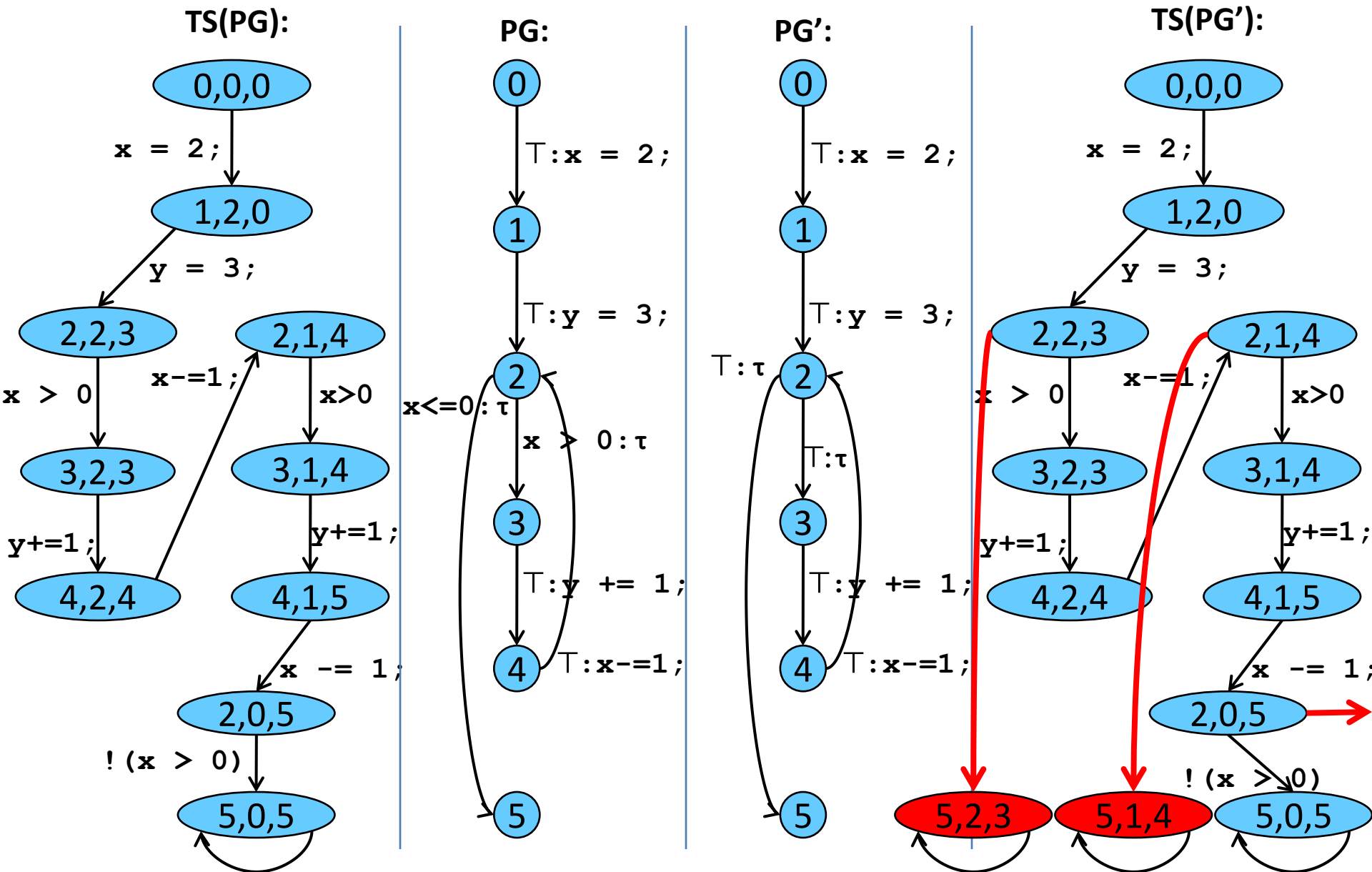
Абстракция графов программ

Пример некорректной абстракции



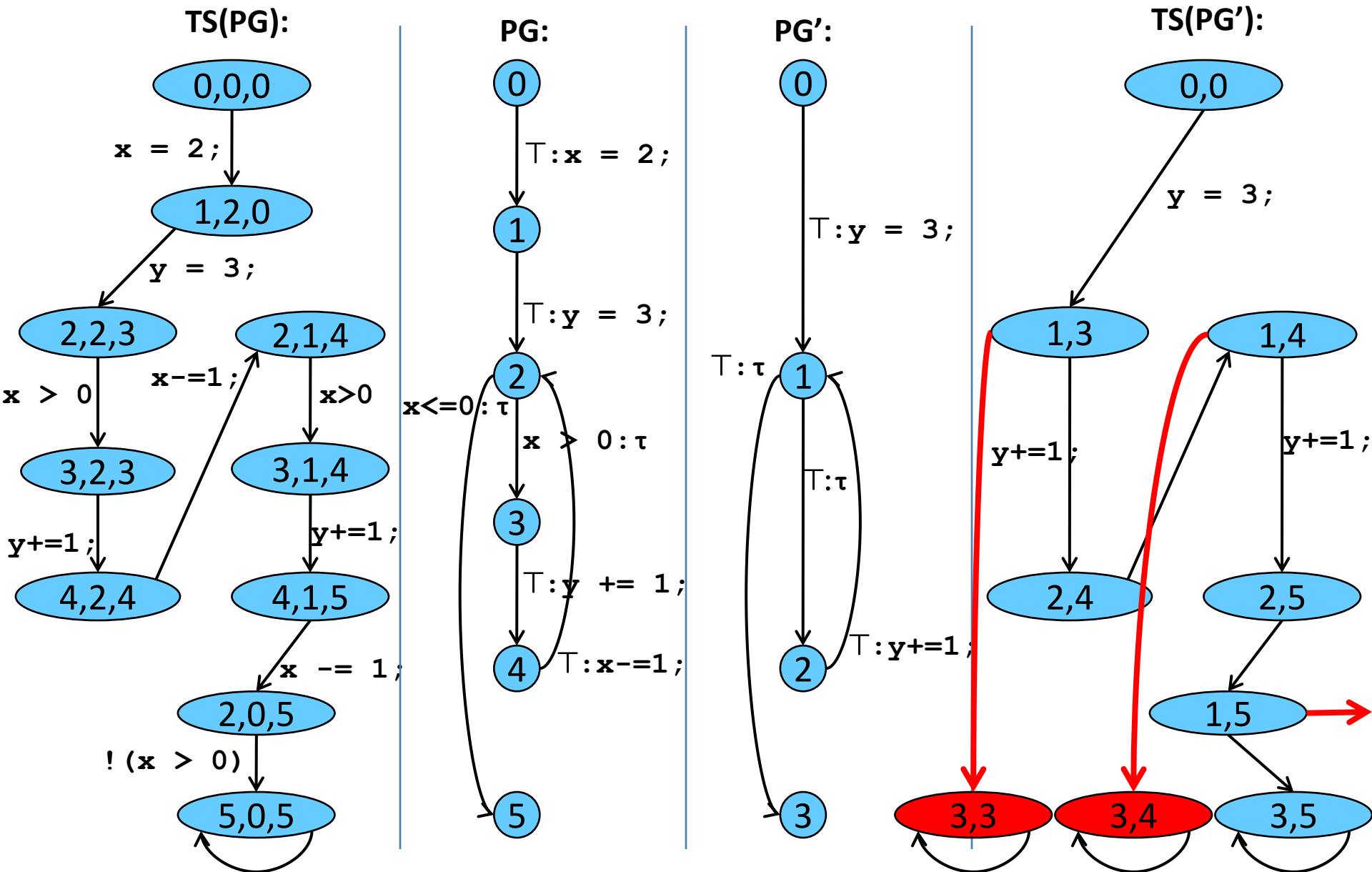
Абстракция графов программ

Пример корректной абстракции



Абстракция графов программ

Пример корректной абстракции



Неформальное определение

$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

$$PG' = \langle Loc', Act', Effect', \rightarrow', Loc_0', g_0' \rangle$$

Будем говорить, что **PG'** моделирует **PG**, если

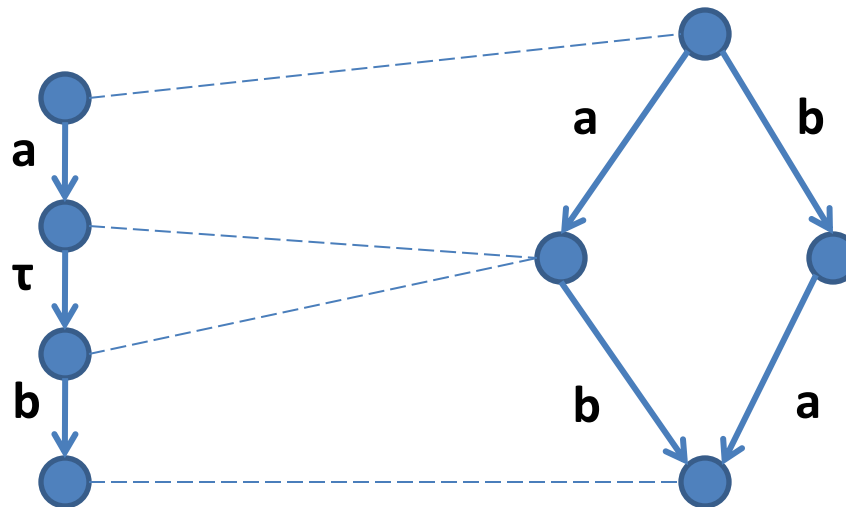
1. В **PG'** присутствуют переменные, соответствующие наблюдаемым переменным **PG**,
2. Все действия **PG**, влияющие на наблюдаемые переменные, отражены в модели,
3. Модель корректно воспроизводит возможные последовательности изменения значений наблюдаемых переменных **PG**.

Отношение (слабого) моделирования (слабой симуляции)

- Будем говорить, что точки l_0 и l'_0 связаны отношением слабого моделирования S , таким, что $(l_0, l'_0) \in S$ тогда и только тогда:

$$\forall l_0 \xrightarrow{g:a} l_1 : ((a \neq \tau) \Rightarrow (\exists l'_1, l'_0 \xrightarrow{g:a} l'_1)) \wedge (l_1, \alpha_{Loc}(l'_1)) \in S$$

Слабая симуляция!



Достаточное условие корректности

- Будем говорить, что PG' моделирует PG , если

$$\exists \alpha_{Loc} : Loc \rightarrow Loc', Loc_0' = \alpha(Loc_0)$$

$$g_0' = g_0 \upharpoonright_{Var_{PG'}}$$

$$\exists \alpha_{Act} : Act \rightarrow Act' \cup \{\tau\}$$

$$\exists \alpha_{Var} : Var_{PG} \rightarrow Var_{PG'} \cup \{\varepsilon\}$$

$$\forall a \in Act, v \in Var_{PG}, (\alpha_{Act}(a) = \tau \wedge \alpha_{Var}(v) \neq \varepsilon) \Rightarrow Effect(a, v) = v$$

$$\forall a \in Act, v \in Var_{PG}, (\alpha_{Act}(a) \neq \tau \wedge \alpha_{Var}(v) \neq \varepsilon) \Rightarrow$$

$$\Rightarrow Effect'(\alpha_{Act}(a), \alpha_{Var}(v)) = Effect(a, v) = v$$

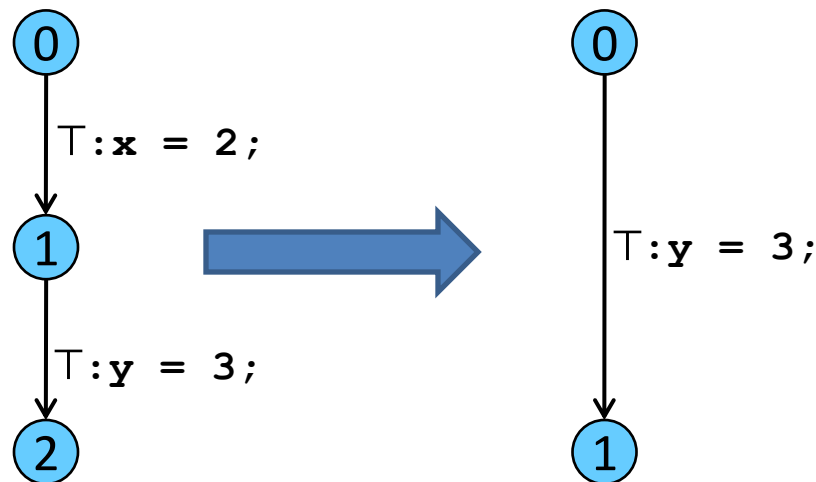
$$\forall a \in Act' \setminus Act, v \in Var_{PG} \cap Var_{PG'}, (Effect'(a, v) = v)$$

а точки Loc_0 и $\alpha(Loc_0')$ связаны отношением слабой симуляции S :

$$\forall l_0 \xrightarrow{g:a} l_1 : ((\alpha_{Act}(a) \neq \tau) \Rightarrow (\exists l_1', l_0' \xrightarrow{g:a} l_1')) \wedge (l_1, \alpha_{Loc}(l_1')) \in S$$

Слабое моделирование

- Отношение слабого моделирования не сохраняет количество шагов между состояниями:

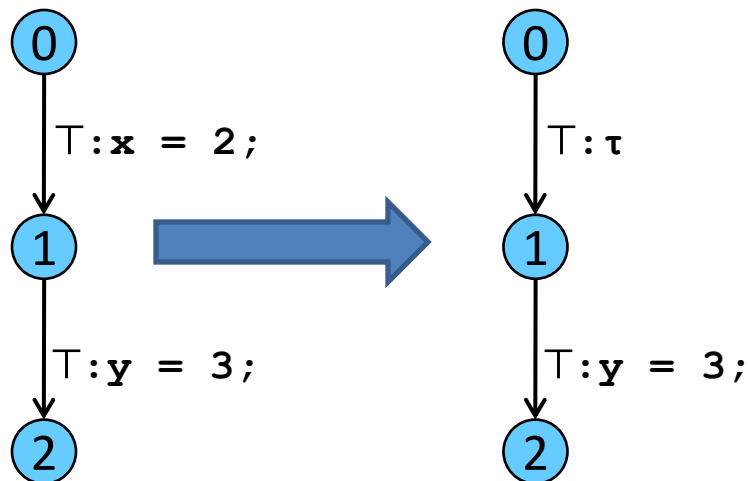


- Не сохраняются свойства, не инвариантные к прореживанию
(LTL: оператор neXt)

Сильное моделирование

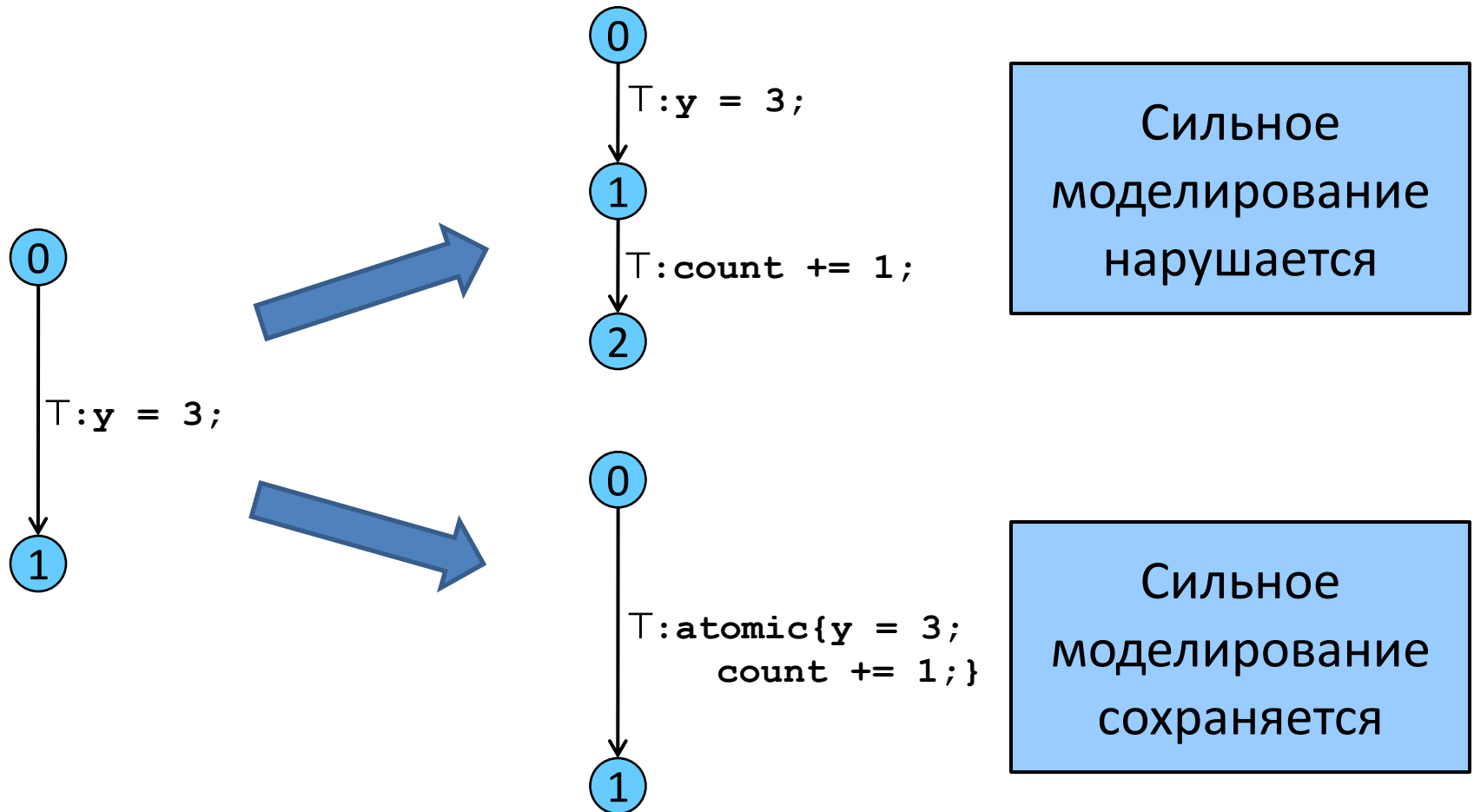
- Для сохранения отношения сильного моделирования необходимо сохранять количество шагов программы между изменением состояния

наблюдаемых
переменных



Сильное моделирование

- С добавлением переменных и операторов в модели тоже не всё просто:



Что дальше

- Будет ещё две лекции (И. Коннов):
 - алгоритмы верификации в SPIN,
 - хранение пространств состояний.
- Сегодня (12 апреля) будет выслано ещё одно задание практикума
 - проверка свойств модели из задания №3,
 - дедлайн 26 апреля.
- 28 апреля – выставление зачёта по практикуму для АСВК (с приёмом последнего задания).
- Экзамен – 7-11 июня
 - перед экзаменом будет консультация.

Оценки по курсу

- Оценка за курс:
 - Оценка по практикуму – 2 = 0..3 балла +
 - Оценка за экзамен (0..3 балла) +
 - Оценка за летучки (-1..1 балл).
- «Слитые» контрольные – в виде доп. вопросов на экзамене.
- Автоматы **будут**
 - Критерии пока не определены.

Спасибо за внимание!
Вопросы?

