НАДЁЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лекция 5:
Логика линейного времени(LTL) (продолжение)
Логики CTL и TCTL
Средство верификации UPPAAL

ВМиК МГУ им. М.В. Ломоносова,
Кафедра АСВК, Лаборатория Вычислительных Комплексов
к.ф.-м.н., доцент Волканов Д.Ю.

План лекции

- Логики CTL, CTL* и TCTL
- Общее описание средства UPPAAL
- Модуль описания
  - Пример системы реального времени
  - Описание сети автоматов
  - Синтаксис выражений
- Модуль симуляции
- Модуль верификации
  - Общее описание
  - Язык запросов

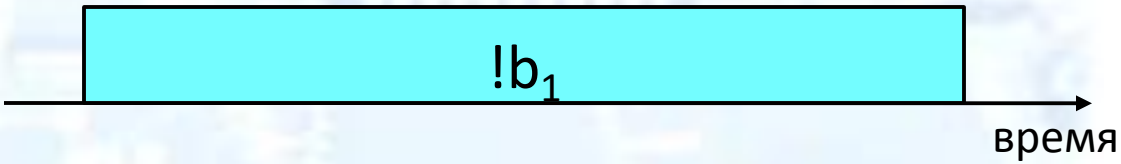
Примеры темпоральных свойств

- **p** всегда истинно; `[]p`
- **p** рано или поздно станет всегда ложным; `<>[]!p`
- **p** всегда рано или поздно станет ложным хотя бы ещё один раз; `[]<>!p`
- **p** всегда ведёт к **¬q**; `[](p->!q)`
- **p** всегда ведёт к тому, что рано или поздно станет истинным **q**. `[](p-> <>q)`

Правильная интерпретация формул LTL

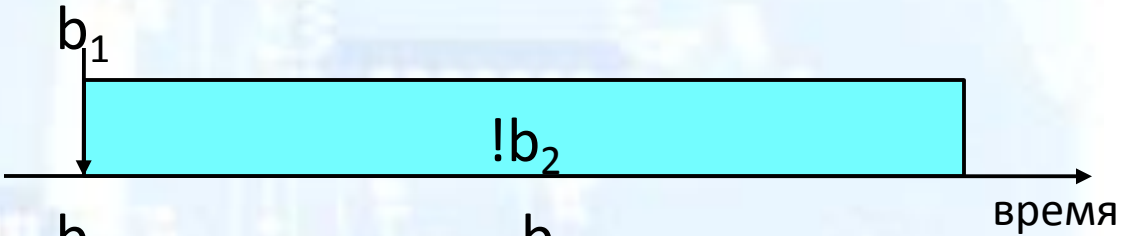
$$LTL:(<>(b_1 \wedge (!b_2 U b_2)) \rightarrow []!a_3$$

1. Пусть b_1 всегда ложно, $p \rightarrow q$ означает, что $!p \vee q$; **формула выполняется.**

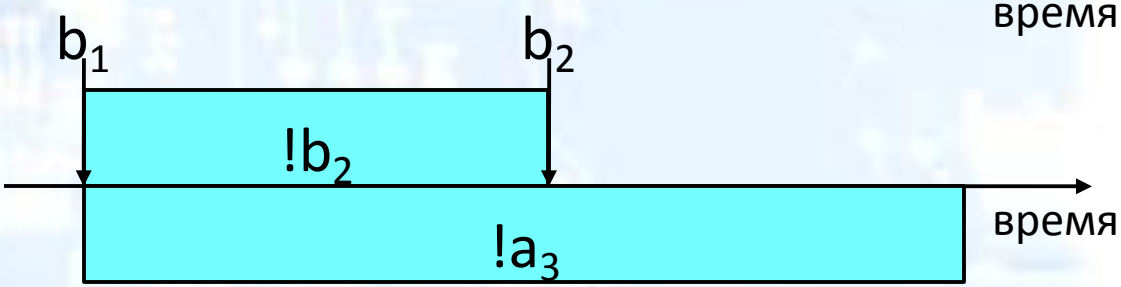


2. Пусть b_1 стало истинно,

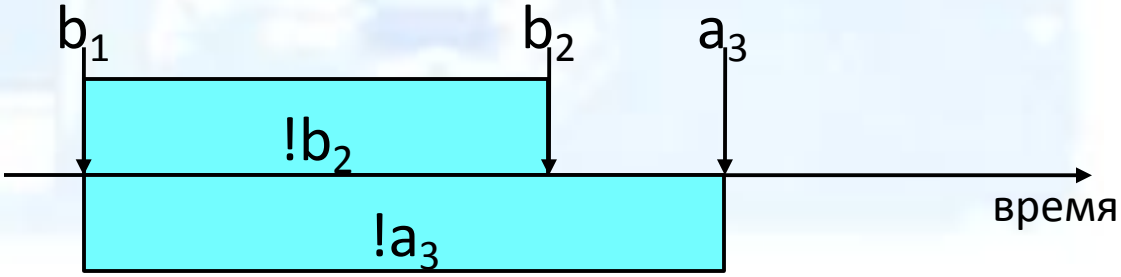
формула выполняется.



3. Пусть b_1 стало истинно, затем – b_2, однако a_3 всегда ложно; **формула выполняется.**



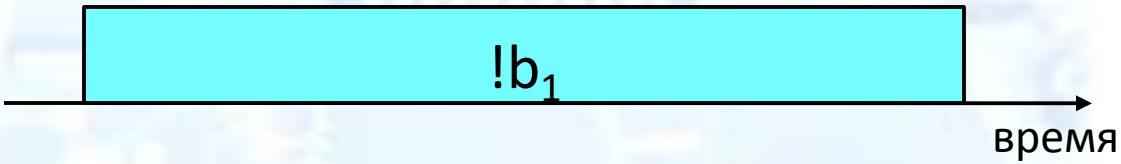
4. Пусть b_1 стало истинно, затем – b_2, затем – a_3; **формула не выполняется.**



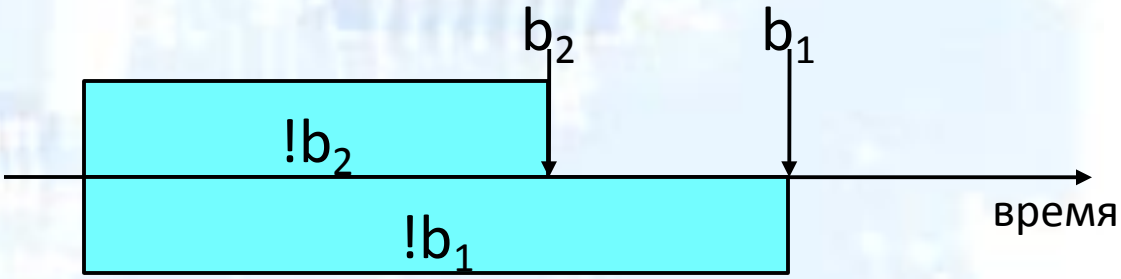
Правильная интерпретация формул LTL

LTL: $(<>b_1) \rightarrow (<>b_2)$

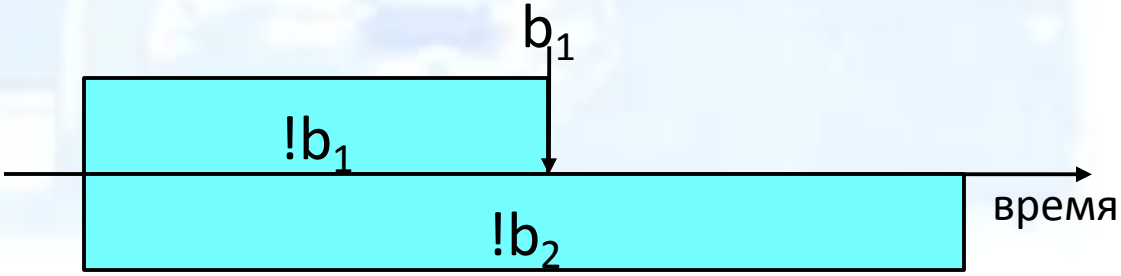
1. Пусть b_1 и b_2 всегда ложно; **формула выполняется.**



2. Пусть и b_1, и b_2 становятся истинными; **формула выполняется.**



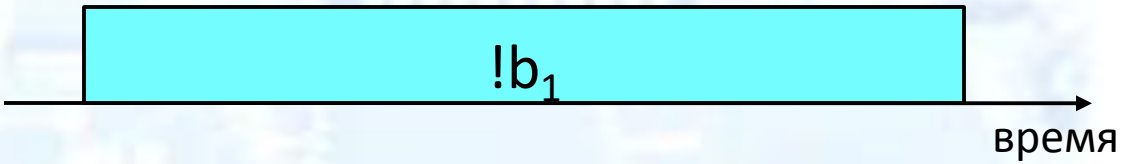
3. Пусть b_1 становится истинным, но b_2 всегда ложно; **формула не выполняется.**



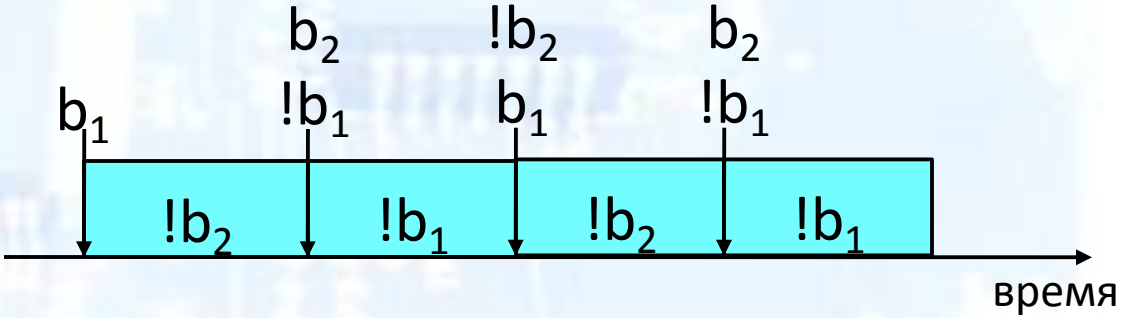
Правильная интерпретация формул LTL

LTL: $[]\left(<>b_1\right)\rightarrow\left(<>b_2\right)$

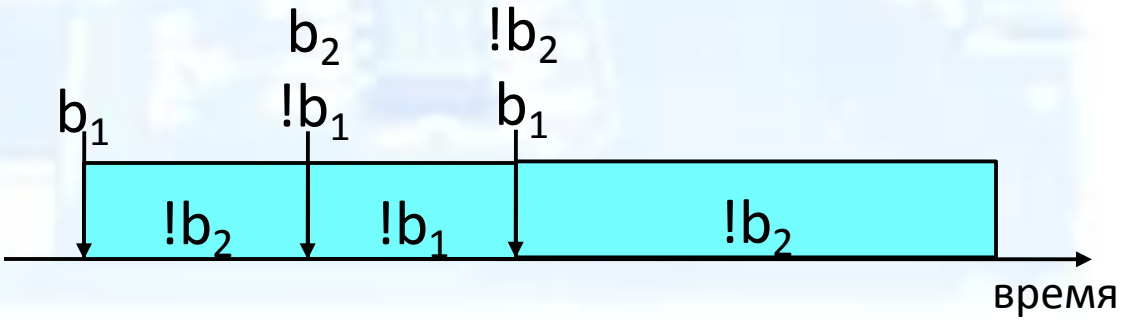
1. Пусть b_1 и b_2 всегда ложно; **формула выполняется.**



2. Пусть и b_1, и b_2 бесконечно чередуются; **формула выполняется.**



3. Пусть b_2 становится истинным только один раз; **формула не выполняется.**



Описание требований при помощи LTL

*“**p** приведёт к **q**”*

- **p -> q**
  - нет темпоральных операторов, т.е. применяется только к первому состоянию;
  - выполняется только если !p$\vee$q в первом состоянии, остальная трасса не рассматривается;
  - **не подходит;**
  - **нужно использовать темпоральные операторы.**

Описание требований при помощи LTL

*“**p** приведёт к **q**”*

- **[] p -> q**

–**правила приоритета!** [] применяется только к p;

–означает ([]p) -> q;

–**не подходит;**

–**нужно расставить скобки.**

Описание требований при помощи LTL

*“**p** приведёт к **q**”*

- **[] (p -> q)**

–проверяем условие во всех состояниях, но причинно-следственная связь между **p** и **q** отсутствует;

–выполняется, только если ! p∨q во всех состояниях;

–не подходит;

–нужно описать, что p является причиной q.

Описание требований при помощи LTL

*“**p** приведёт к **q**”*

- **[] (p -> <>q)**

–уже лучше;

–тем не менее, формула выполнима, если q становится истинным в том же состоянии, что и p – причинно-следственная связь отсутствует;

–не подходит;

–нужно описать, что q не может произойти раньше следующего шага после p.

Описание требований при помощи LTL

*“**p** приведёт к **q**”*

- **[] (p -> X(<>q))**

–практически то, что нужно;

–формула выполнима, если p всегда ложно;

–возможно, не подходит;

–нужно описать, что p обязательно произойдёт и приведёт к q.

Описание требований при помощи LTL

*“**p** приведёт к **q**”*

- **[] (p -> X(<>q)) && (<>p)**

–скорее всего, мы имели ввиду именно это;

–несколько отличается от первоначального p->q;

–LTL позволяет выразить множество различных оттенков свойства;

–подойдёт ли такое свойство для модели параллельной программы?

Оператор ne**X**t

- Оператор **X** нужно использовать аккуратно:
  - с его помощью делается утверждение о выполнимости формулы на непосредственных потомках текущего состояния;
  - в распределённых системах значение оператора X неочевидно;
  - поскольку алгоритм планирования процессов неизвестен, не стоит формулировать спецификацию в предположении о том, какое состояние будет следующим;
  - стоит ограничиться предположением о справедливости планирования.

Свойства, инвариантные к прореживанию

- Пусть φ – трасса некоторого вычисления над пропозициональными формулами P,

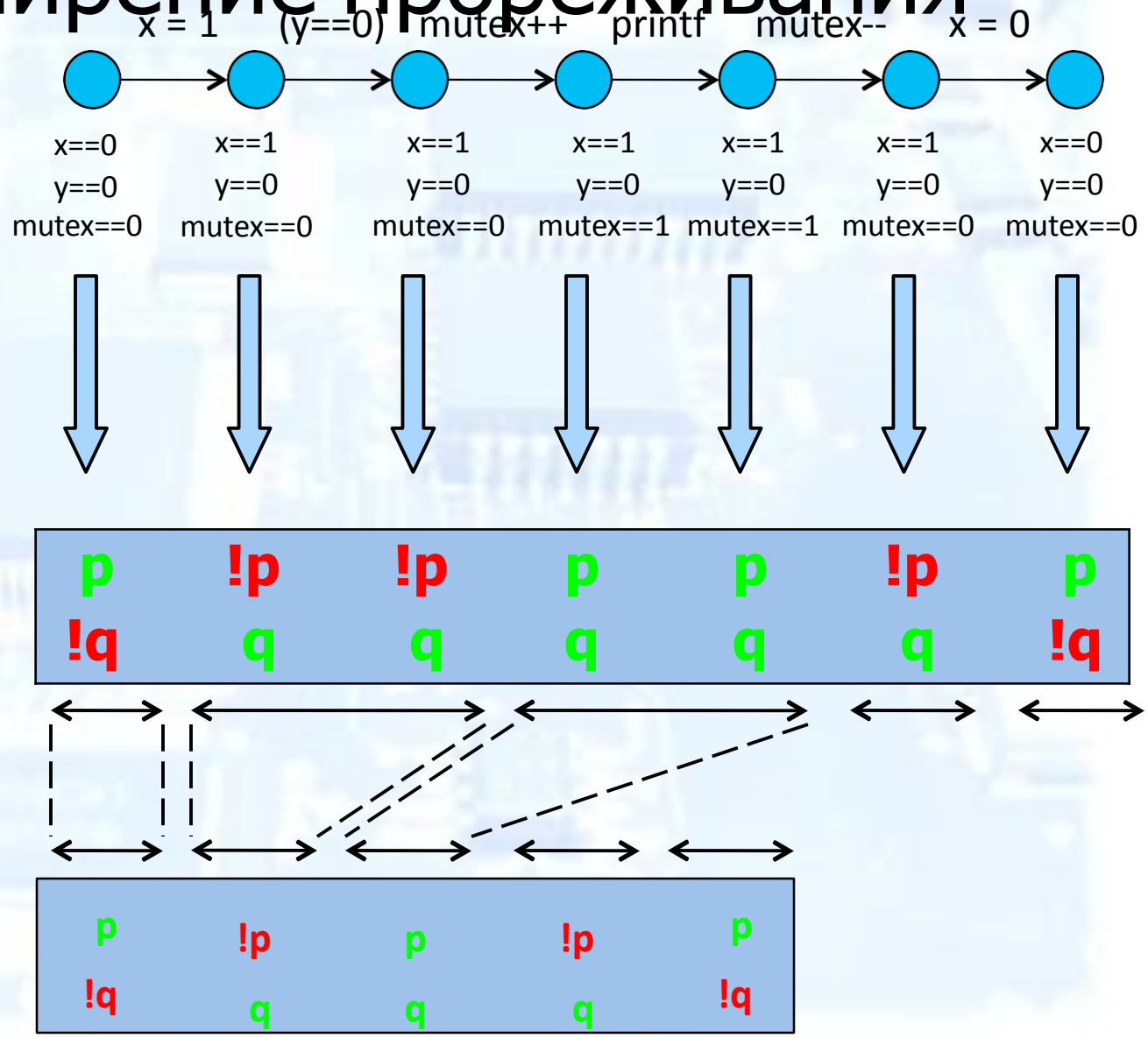
– по трассе можно определить, выполняется ли на ней темпоральная формула,

– трассу можно записать в виде $\varphi = \varphi_1^{n1}\ \varphi_2^{n2}\ \varphi_3^{n3}\ \cdots$, где значения пропозициональных формул на каждом интервале совпадают.

•Обозначим E(φ) набор всех трасс, отличающихся лишь значениями n1, n2, n3 (т.е. длиной интервалов)

– E(φ) называется **расширением прореживания** φ.

Расширение прореживания



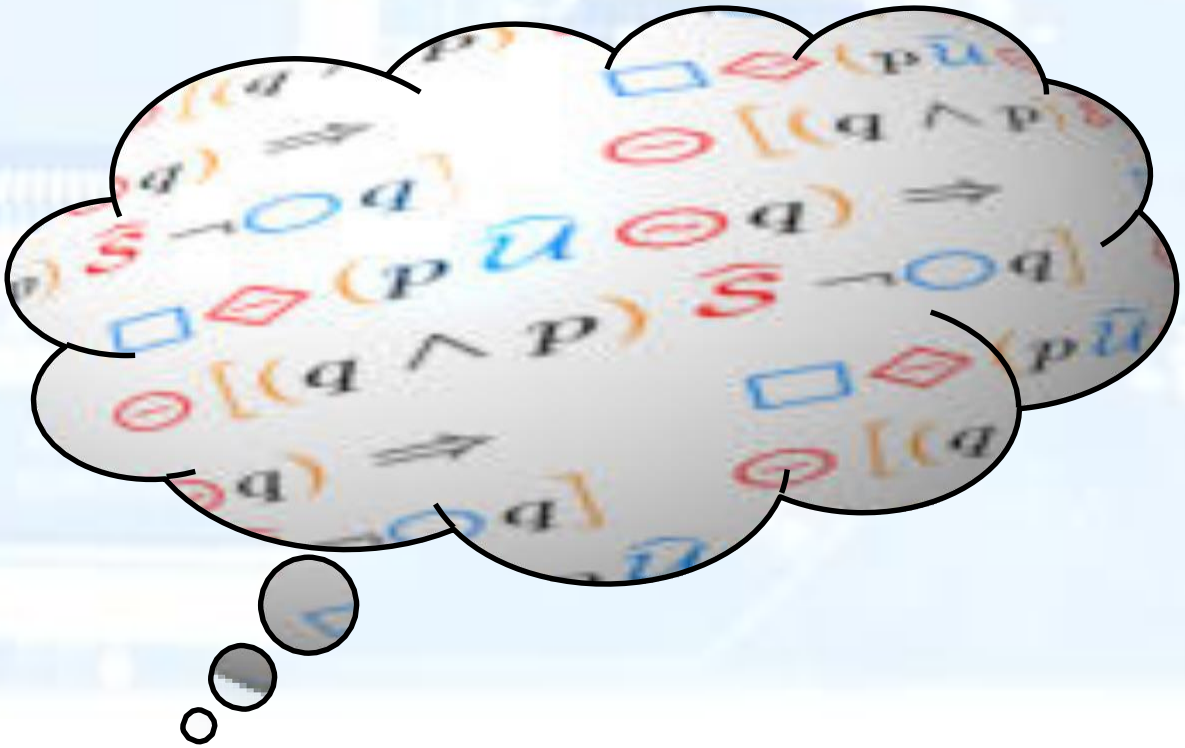
Свойства, инвариантные к прореживанию

- Свойство φ, инвариантное к прореживанию, либо истинно для всех трасс из E(φ), либо ни для одной из них:

$$\varphi \models f \quad \rightarrow \forall v \in E(\varphi), v \models f$$

- истинность такого свойства зависит от порядка, в котором пропозициональные формулы меняют свои значения, и не зависит от длины трассы;
- **Теорема**: **все** формулы LTL без оператора X инвариантны к прореживанию.
- Более того, в рамках LTL без X можно описать **все** свойства, инвариантные к прореживанию.

Практические приёмы описания свойств на LTL



Практические приёмы описания свойств на LTL

- Выполнимость формулы LTL проверяется только для первого состояния в трассе
- Темпоральные операторы управляют проверкой выполнимости своих аргументов
- Сложное свойство можно (и нужно!) строить как суперпозицию простых
- Суперпозиция темпоральных операторов не ограничивает диапазон действия «вложенного» оператора.

Выполнимость формул LTL

- Выполнимость формулы LTL определена и проверяется для одного (первого) состояния трассы
- Распространение свойств на другие состояния управляется темпоральными операторами
- Единственный оператор, который может ограничить сверху проверку выполнимости формулы – Until

выполнимость подформул – уже нет

для этого они и нужны

описать свойство, выполняющееся для участка программы (одной итерации, одного вызова функции) без Until не получится

Суперпозиция формул LTL

- Составлять сложные формулы LTL нужно методом суперпозиции простых формул
- Внешняя формула задаёт, на каких участках вычислений будет проверятся подформула
- Подформула задаёт свойства, проверяемые для участков вычислений
- Суперпозиция темпоральных операторов не ограничивает диапазон действия «вложенного» оператора.

База шаблонов темпоральной логики http://patterns. projects.cis.ksu.edu

База шаблонов темпоральной логики http://patterns. projects.cis.ksu.edu

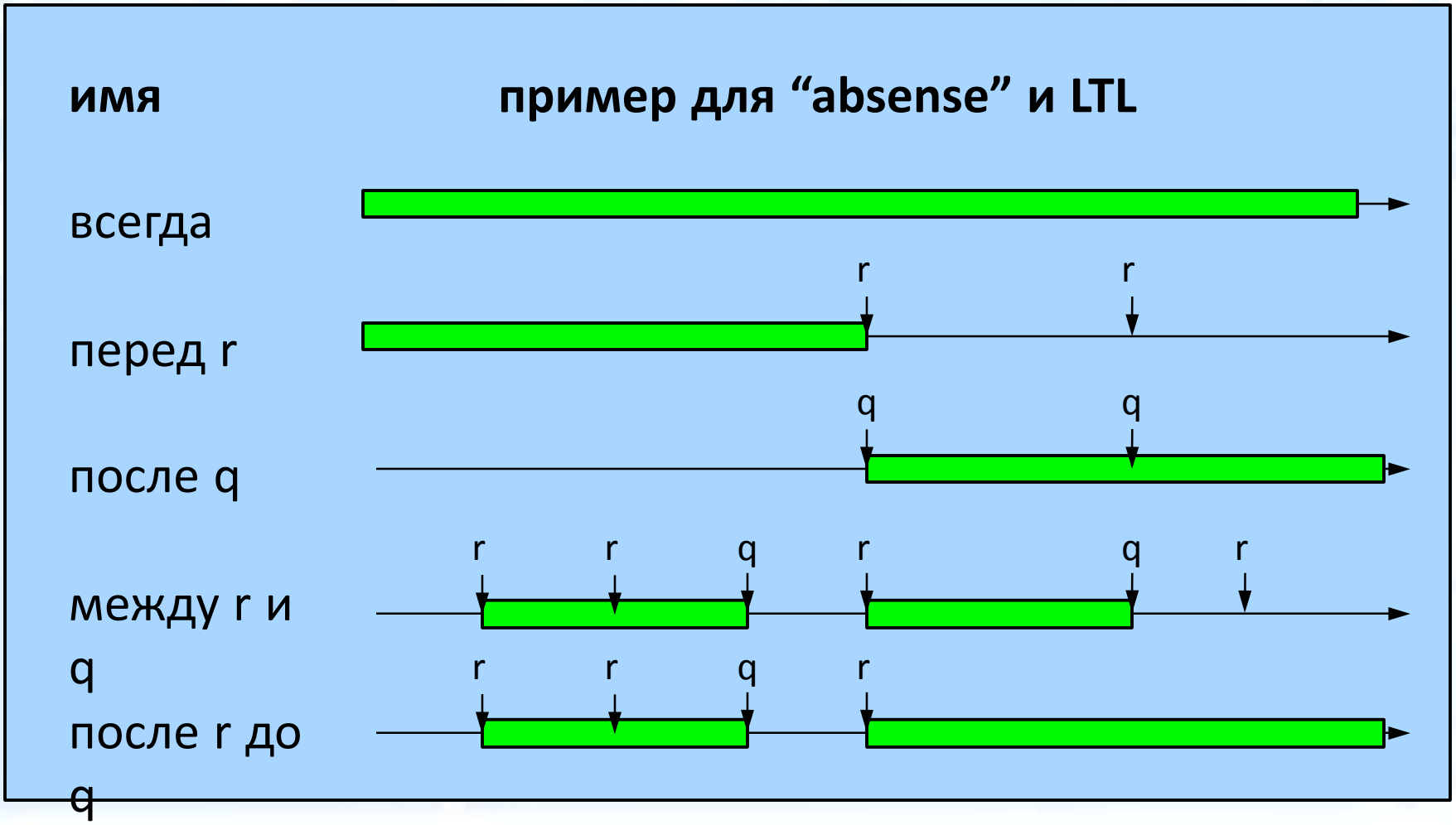
- Для каждого шаблона – пять вариантов формул:

имя	пример для “absense” и LTL
всегда	`[](!p)`
перед r	`<>r -> (!p U r)`
после q	`[](q -> [](!p))`
между r и q	`[]((r && !q && <>q) -> (!p U q))`
после r до q	`[]((r && !q) -> ((!p U q) || []!p))`

В чём разница?

База шаблонов темпоральной логики http://patterns.projects.cis.ksu.edu

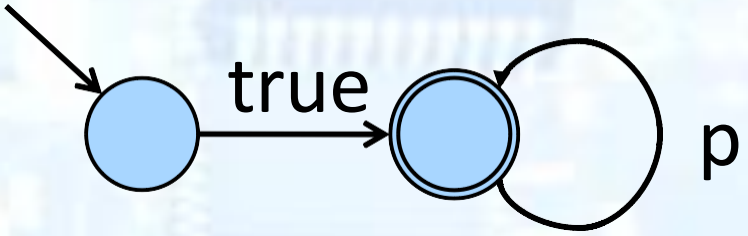
- Для каждого шаблона – пять вариантов формул:



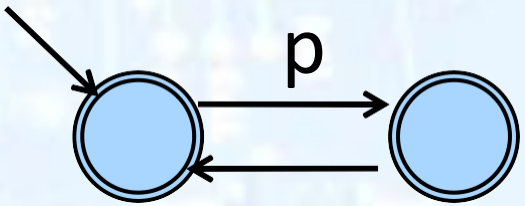
Пример свойства, не выразимого на LTL

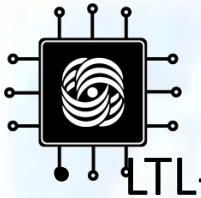
- (p) **может** быть истинным после выполнения системой чётного числа шагов, но **никогда не истинно** после **нечётного**.

- `[]X(p)` не подходит



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





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
U	состоянии сильный unil
U	слабый unil
<>	рано или поздно
[]	всегда

Серым отмечены операторы, которые можно вывести из других:

φ_1 || φ_2	==	!(! φ_1 && ! φ_2)
<> φ	==	true U φ
[] φ	==	!<> !φ
φ_1 *U* fi2	==	[] φ_1 || (φ_1 U φ_2)

грамматика:

пропозициональные формулы:

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

темпоральные формулы:

- f
- ! φ
- (φ)
- φ && φ
- φ || φ
- X φ
- φ U φ
- <> φ
- [] φ

p – некоторый пропозициональный символ

f – некоторая пропозициональная формула

φ – некоторая темпоральная формула

Логика CTL*

Логика ветвящегося времени:

- использует кванторы ∀ и ∃,
- использует F вместо <> и G вместо [].

Операторы

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

Серым отмечены операторы, которые можно вывести из других:

$\varphi_1 \,||\, \varphi_2$ = $!(!\varphi_1 \,\&\&\, !\varphi_2)$

= $!E\,!\varphi$

$= A\,\varphi$ = $true\ U\ \varphi$

= $!F!\varphi$

формулы состояния: p, !f, (f), f && f, f || f, A φ, E φ

формулы пути: f, ! Φ, (φ), φ && φ, φ || φ, X φ, φ U φ, 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.

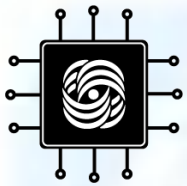
Корректная CTL формула:

p
! φ
φ && φ
φ || φ
E X φ
E (φ U φ)
A (φ U φ)

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 **<>p** означает:

A<>p **для всех** вычислений, начинающихся в исходном состоянии s_0, выполняется **<>p**

В CTL можно выразить:

EF(p) **существует** вычисление, для которого выполняется **<>p**

AF(p) **для всех** вычислений выполняется **<>p**

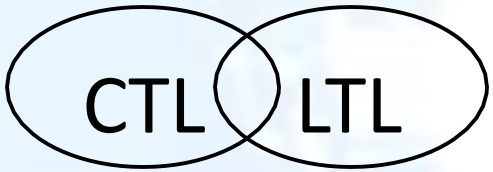
AG(p) **для всех** вычислений **p** – инвариант

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

итд.

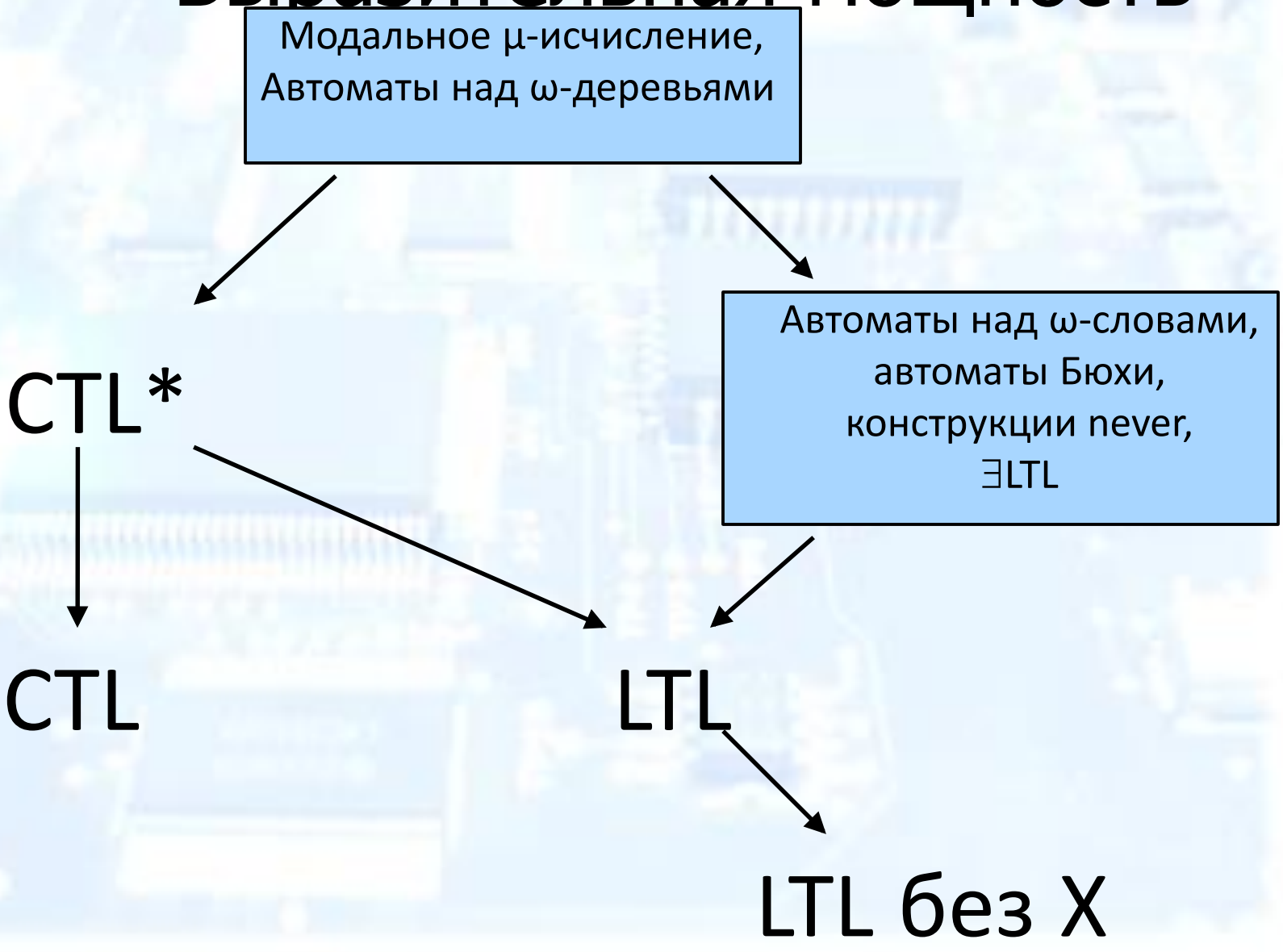
Выразительные возможности CTL* и CTL

- 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 []<>(p),
    - при помощи []<>(p) в LTL задаются ограничения справедливости;



  - на CTL можно описать свойства, не выразимые на LTL:
    - на LTL нельзя описать свойства вида AGEF(p),
    - AGEF(p) используется для описания свойства reset: из любого состояния система может перейти в нормальное.

Выразительная мощность



Примеры формализаций высказываний

- *Джейн вышла замуж и родила ребенка*

P(Джейн_выходит_замуж /\ F Джейн_рожает_ребенка)

- *Джейн родила ребенка и вышла замуж*

P(Джейн_рожает_ребенка /\ F Джейн_выходит_замуж)

- *Джон умер и его похоронили*

P(Джон_умирает /\ XF Джона_хоронят)

- *Если я видел ее раньше, то я ее узнаю при встрече*

G(P Увидел $\Rightarrow$ G(Встретил $\Rightarrow$ X Узнал))

- *Ленин – жил, Ленин – жив, Ленин – будет жить* (В.В.Маяковский)

PG Ленин_жив

- *Любое посланное сообщение будет получено*

G(Послано(m) $\Rightarrow$F Получено(m))

- *Вчера он сказал, что придет завтра, значит, он придет сегодня*

$X^{-1}X$ Приходит $\Rightarrow$ Приходит (истинно)

LTL и CTL – подклассы CTL*

Алгоритмы проверки выполнения формул CTL* сложны ⇒ нужны подклассы

В LTL - формулы пути, которые должны выполняться для всех вычислений предваряются квантором пути A

В CTL каждый темпоральный оператор предваряется квантором пути A или E

Формулы LTL:

AG(p ⇒**F** q)

A (¬ a\/ **G**b & (a**U** ¬c))

A (a **U** ¬b)

Формулы CTL:

AG(p&¬**E**F(q⇒r))

EF(a & **E**(a**U** ¬c))

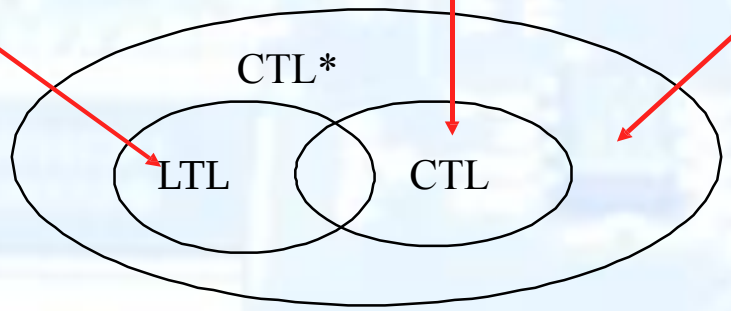
A (a **U** ¬ b)

Формулы CTL*:

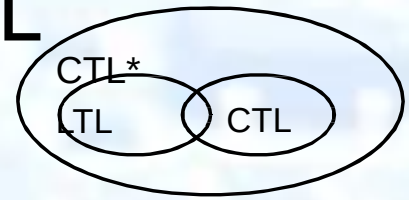
E(¬p & **X** **A** **F** q)

EX (a & **A**X(b**U**c)]

A (a **U** ¬ (**F** b))



Сравнение логик LTL и CTL

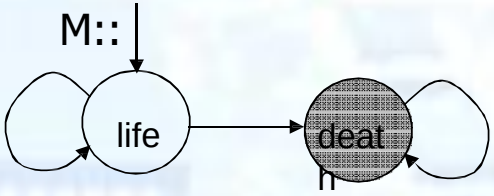


- Формулы этих двух логик характеризуют свойства разных объектов
  - LTL – формулы пути, CTL – формулы состояний
- Выражают свойства вычислений, которые представлены по-разному
  - LTL – множество поведений, CTL – деревья поведений
- Интерпретируются по-разному
  - формулы LTL - на бесконечном множестве поведений
  - формулы CTL – на конечном множестве состояний
- Методы анализа - алгоритмы model checking - совершенно разные
- Выразительная мощь несравнима: есть формулы CTL, невыразимые в LTL, и наоборот
  - например, $FG\varphi$ не может быть выражена в CTL (с некоторого времени в будущем φ будет все время истинным) AFAGp сильнее, AFEGp – слабее
  - например, $AGEF\varphi$ в CTL не может быть выражена в LTL

Пример формализации в CTL*

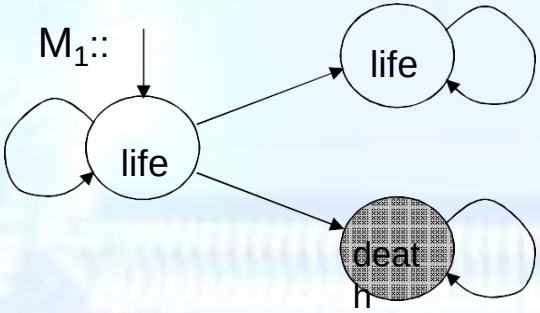
“Летят за днями дни, и каждый час уносит
Частичку бытия, а мы с тобой вдвоем
Предполагаем жить, и глядь — как раз - умрем”

А.С.Пушкин



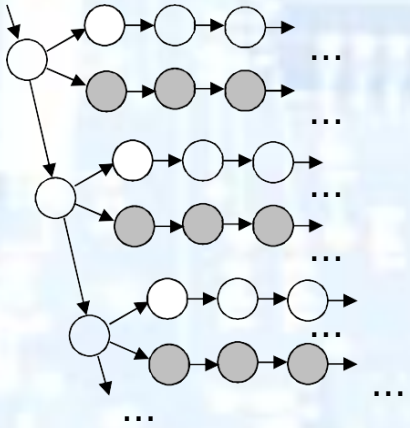
φ_{br} = **A** [(**G** *life*) $\Rightarrow$ (**GEX** *death*)]

M |= φ_{br}

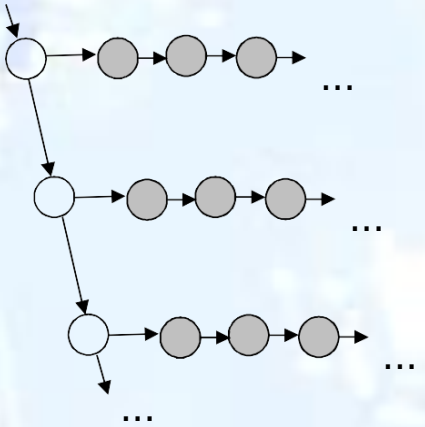


$M_1 \not\models \varphi_{br}$

Развертка M1:

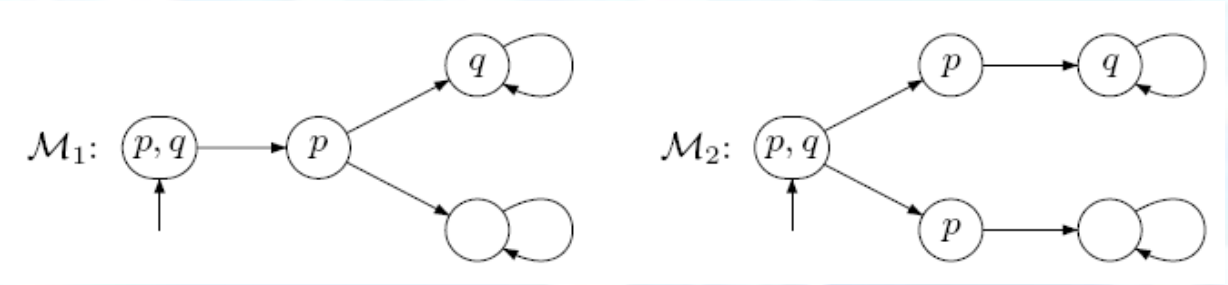


Развертка M:



На развёртке M1 формула φ_{br} не выполняется: ее развертка имеет траектории “вечной жизни” без возможного ответвления на состояния, помеченные *death*

Различия LTL и CTL*



- Эти две модели не различаются LTL, но различаются CTL*:
  - AG(p→EFq). В M1 эта формула выполняется, в M2 она не выполняется
  - В LTL обе модели представляются одним и тем же множеством из двух вычислений, поэтому LTL их не различает

Логика TCTL

TCTL – Timed CTL – естественное расширение операторов U, F, ... логики CTL количественной информацией.

Грамматика TCTL (= CTL + Time):

$\phi ::= p \mid \alpha \mid \neg\phi \mid \phi \wedge \phi \mid z \text{ in } \phi \mid E[\phi \, U \, \phi] \mid A[\phi \, U \, \phi]$

p – атомарный предикат
α - ограничение на таймеры и формульные часы z – формульные часы
z in ϕ - введение новых часов в формулу ϕ
$E[\phi \, U \, \phi]$, $A[\phi \, U \, \phi]$ – как в CTL

Обозначение: $E[\phi \, U_{\alpha} \psi] \equiv z \text{ in } E[(\phi \, \& \, \alpha) U \, \psi]$ Выводимые операторы

$EF_{\alpha}\phi \equiv E[\text{True } U_{\alpha} \, \phi]$ и т.д.

Формулы TCTL включают $E[\phi \, U_{\sim k} \psi]$, $A[\phi \, U_{\sim k} \psi]$, $EF_{\sim k}\phi$, $EG_{\sim k}\phi$, $AF_{\sim k}\phi$, $AG_{\sim k}\phi$
где ~ - любой символ из $\{<, \leq, =, \geq, >\}$ и k – рациональное число

Примеры свойств реального времени

Model checking временных автоматов относительно выполнения заданной формулы TCTL сводится к model checking его регионального графа относительно выполнения формулы CTL

1.[$p\ U_{<2}\ q$] – p истинно непрерывно до тех пор, пока не станет истинно q, и истинность q наступит не позднее, чем через 2 единицы времени

2.AG(problem $\Rightarrow AG_{\geq 5}$ alarm) – как только проблема возникла, сигнал alarm зазвучит сразу и будет звучать не менее 5

3.AG($\neg$far $\Rightarrow AF_{<7}$ far) – поезд покинет область контроля не позже, чем через 7

4.AG [send(m) $\Rightarrow AF_{<5}$ receive(r_m)] – подтверждение приходит в пределах 5

5.EG [send(m) $\Rightarrow AF_{>4}$ receive(r_m)] - подтверждение может быть получено более, чем за 4

6.AG [$AF_{=15}$ tick] – тики следуют периодически точно через 15 е.в. (но, кроме того, могут быть и в промежутках)

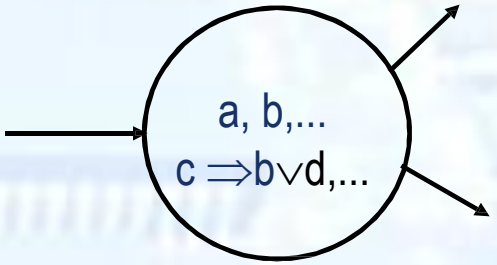
7.AG ($x \leq y$) - таймер x всегда не больше таймера y

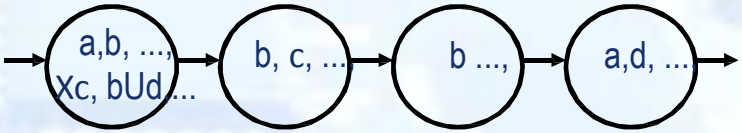
8.A [off U $x \geq 3$] по любому пути из начального состояния если светофор выключен, то он будет выключен до тех пор, пока таймер x не будет иметь значение ≥ 3

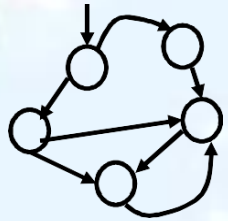
Общие идеи в TL

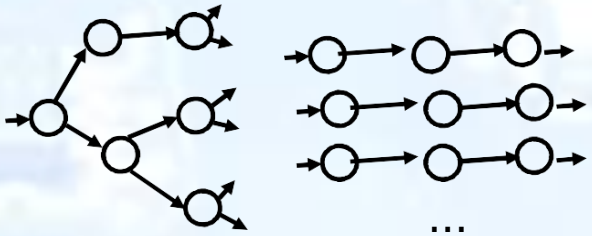
- Логика высказываний строится введением
- В логике линейного времени LTL кроме атомарных утверждений и операций логики высказываний вводятся темпоральные операторы {U, X} (кроме них удобно использовать еще F и G)
- По конкретной цепочке состояний (миров) в каждом состоянии можем вычислить истинностные значения любой формулы темпоральной логики LTL
- В логике CTL* добавляются кванторы пути, позволяющие различать свойства различных путей
- Формула CTL* определена для конкретной интерпретации (структуры Крипке) и всех возможных ее вычислений

- CTL является подмножеством CTL* - в формулах CTL каждый темпоральный оператор предваряется квантором пути









Особенности UPPAAL

UPPAAL – это весьма популярное средство верификации систем реального времени

- Простое в использовании
- Бесплатное для научных исследований
- Имеет в основе простую, но при этом довольно мощную математическую модель
- Малопригодно для разработки больших систем
- Не поддерживает описание иерархии вложенности компонентов систем

Структура

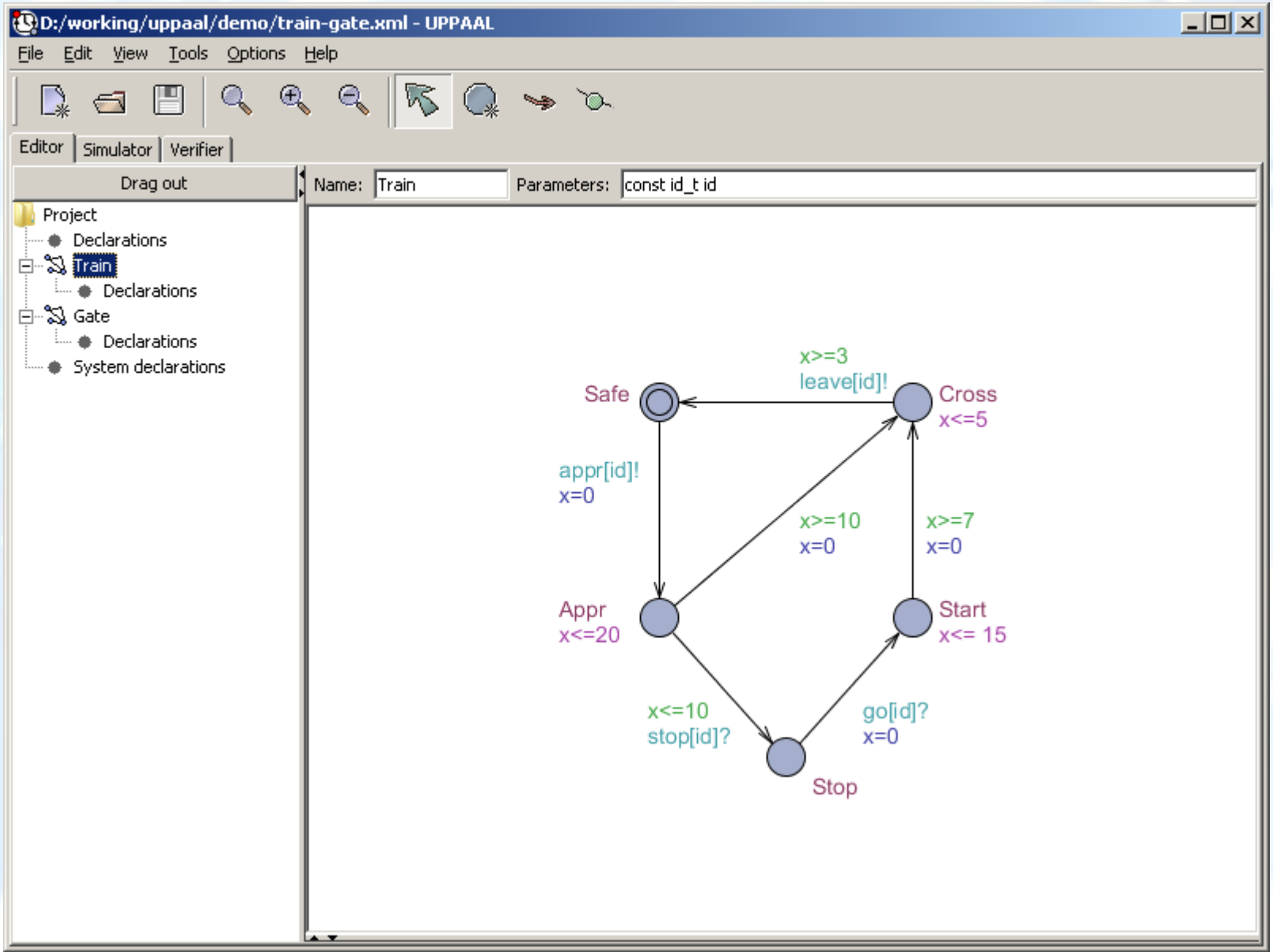
- Модуль описания
  - входная модель – расширение модели сети временных автоматов
  - возможность описания параметризованных шаблонов
  - наличие данных различной степени локальности
- Модуль симуляции
  - генерация и визуализации трассы сети
  - различные типы генерации (случайная, из файла)
- Модуль верификации
  - проверка темпоральных свойств
  - предоставление трассы-контрпримера
  - воспроизведение контрпримера в модуле симуляции

Пример

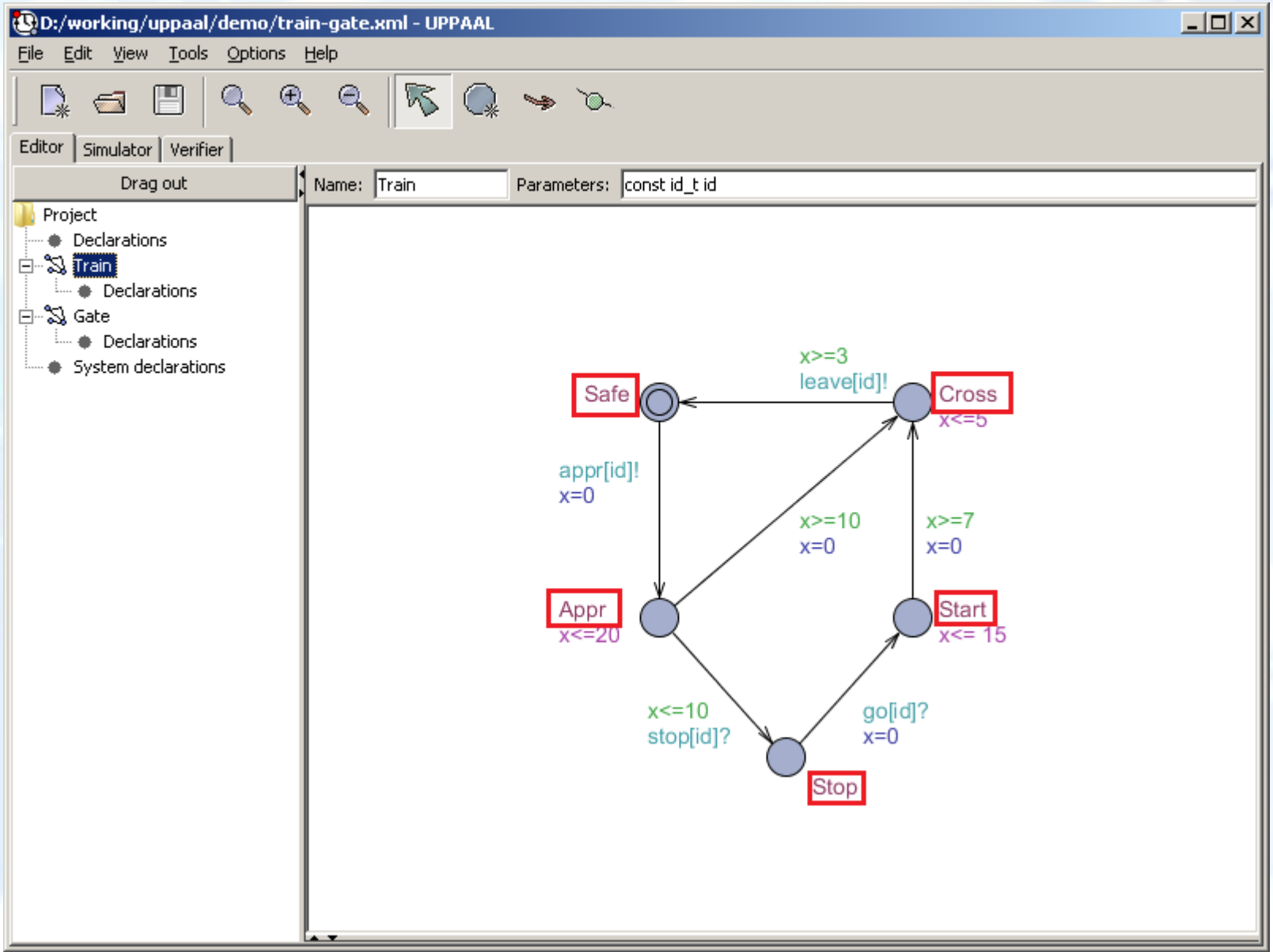
Train Gate

- Система обеспечения доступа к критическому ресурсу
- Требуется обеспечить проход нескольких поездов по одному мосту
- В наличии устройство управления поездами
- Остановка и разгон поезда происходят не мгновенно

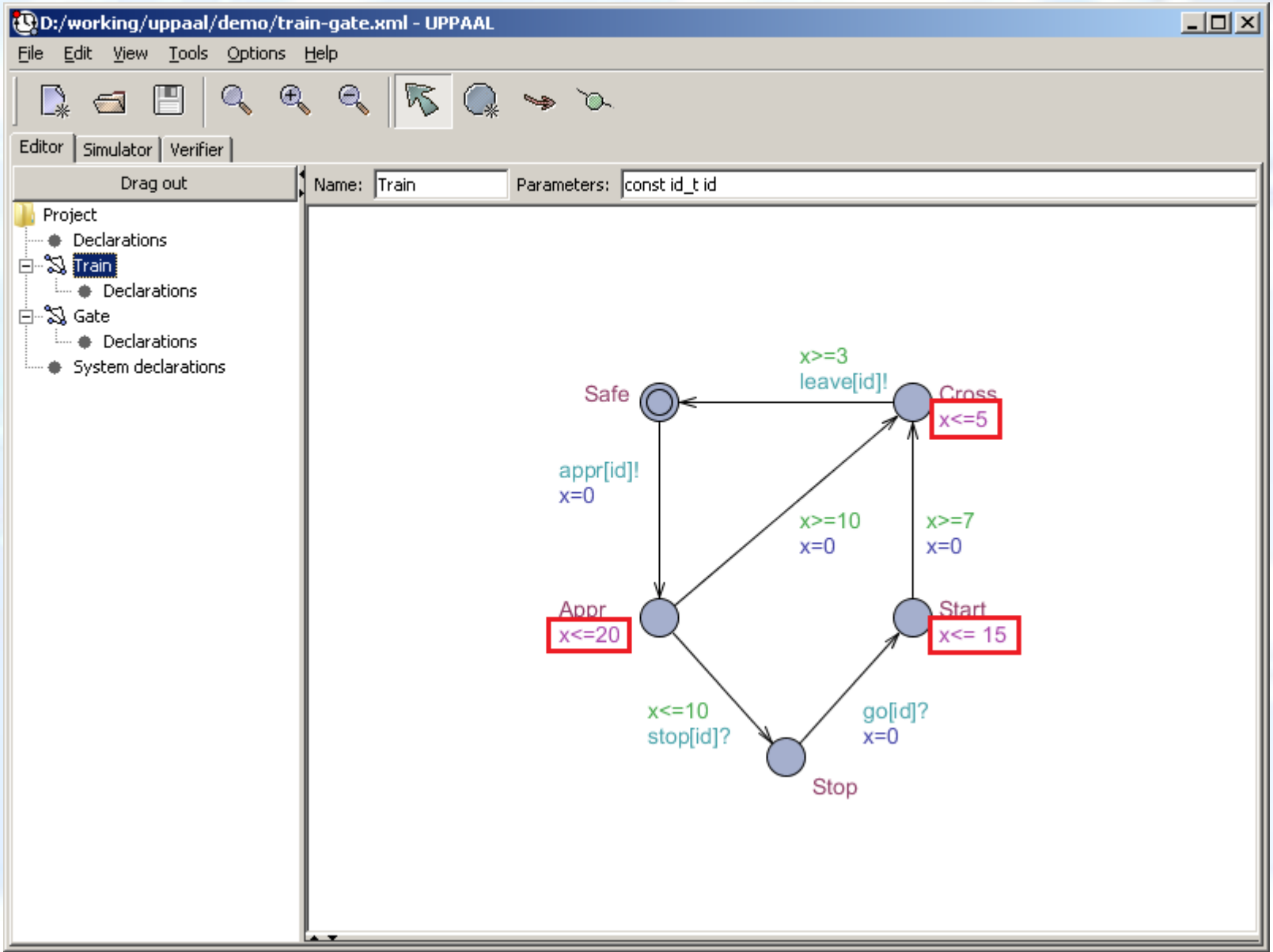
Модуль описания



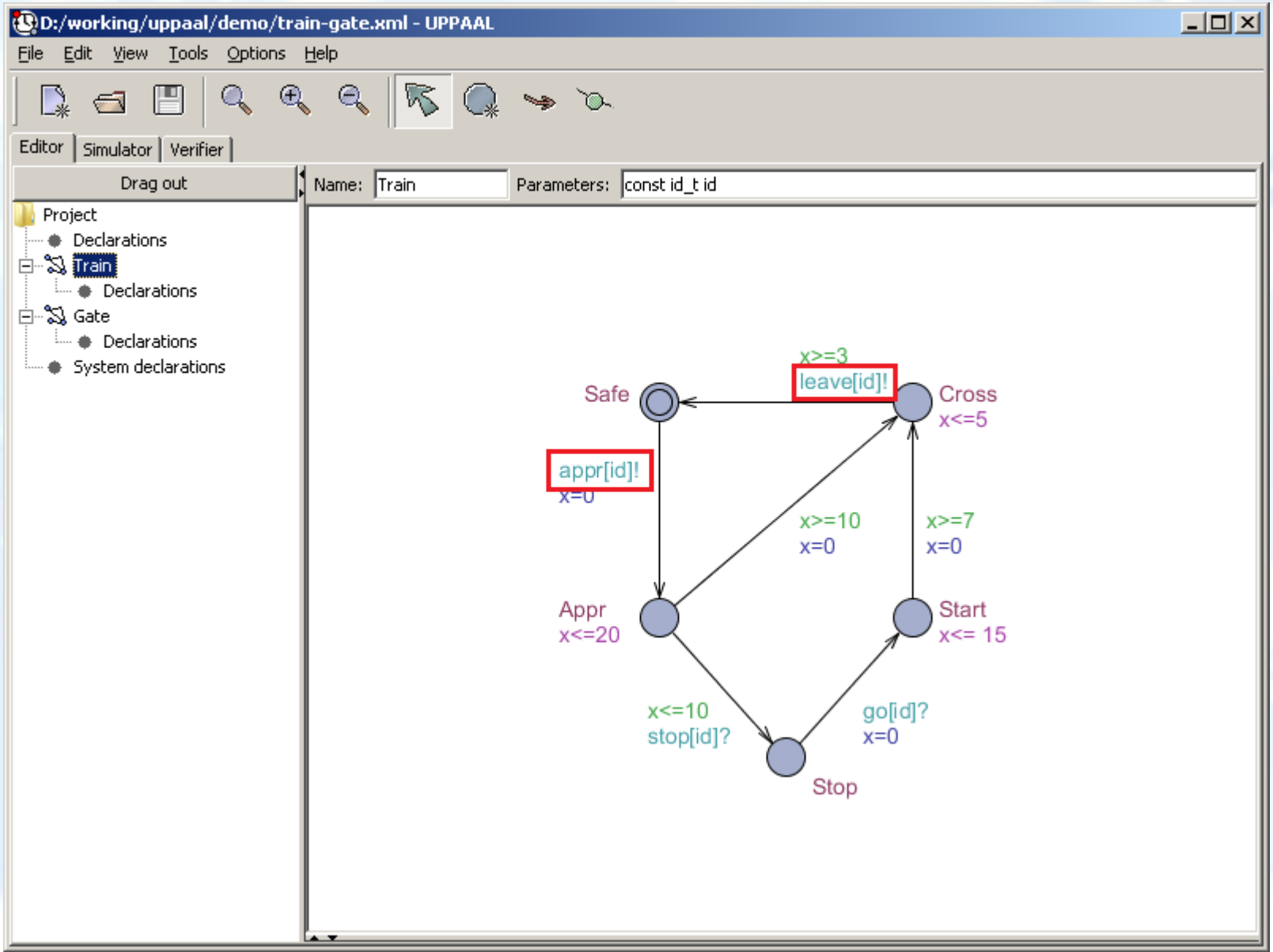
Имена состояний



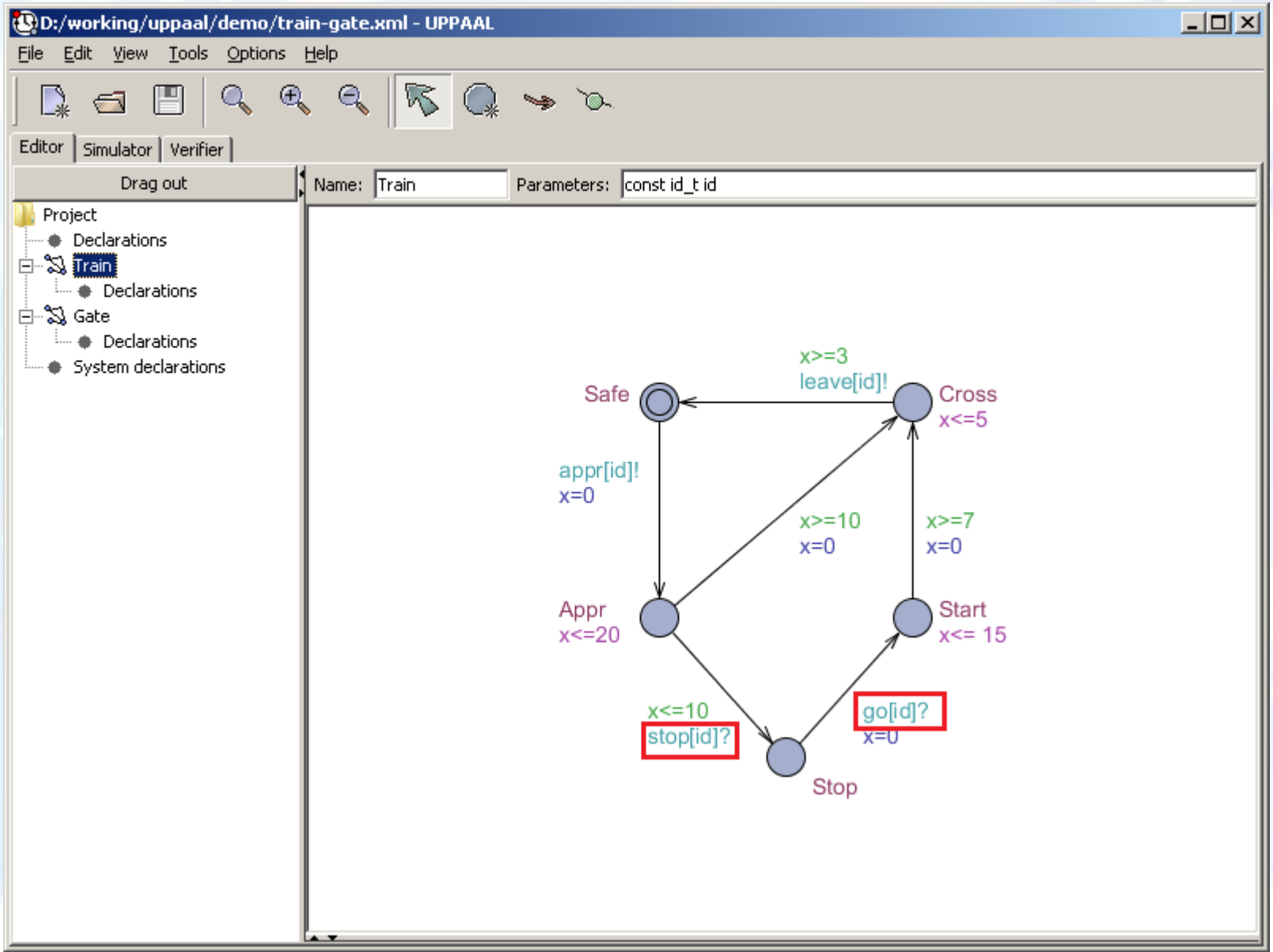
Инварианты


D:/working/uppaal/demo/train-gate.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Project
Declarations
Train
Declarations
Gate
Declarations
System declarations
Name: Train
Parameters: const id_t id
x>=3
leave[id]!
Safe
Cross
x<=5
appr[id]!
x=0
x>=10
x=0
x>=7
x=0
Appr
x<=20
Start
x<= 15
x<=10
stop[id]?
go[id]?
x=0
Stop

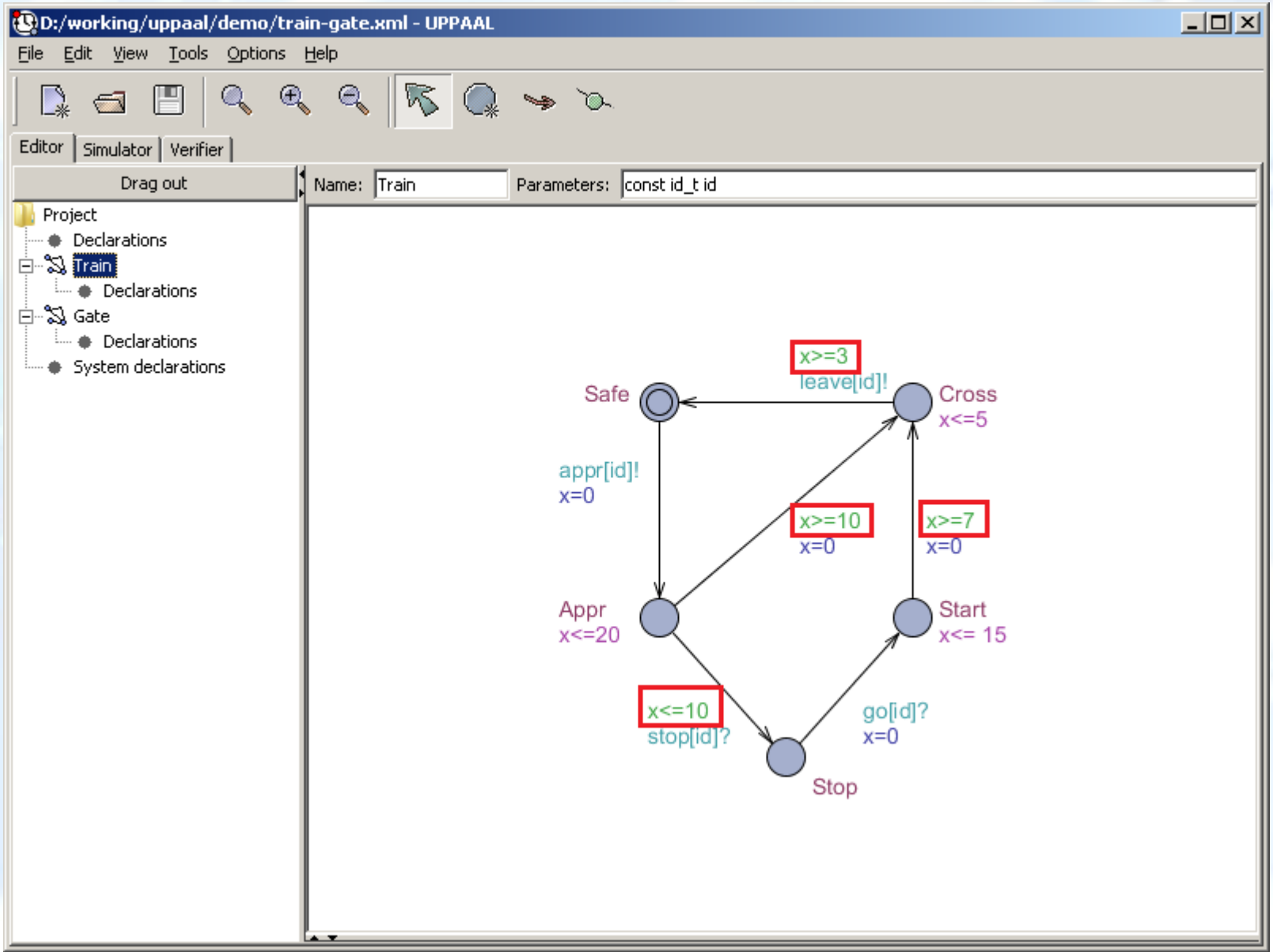
Посылка сигналов



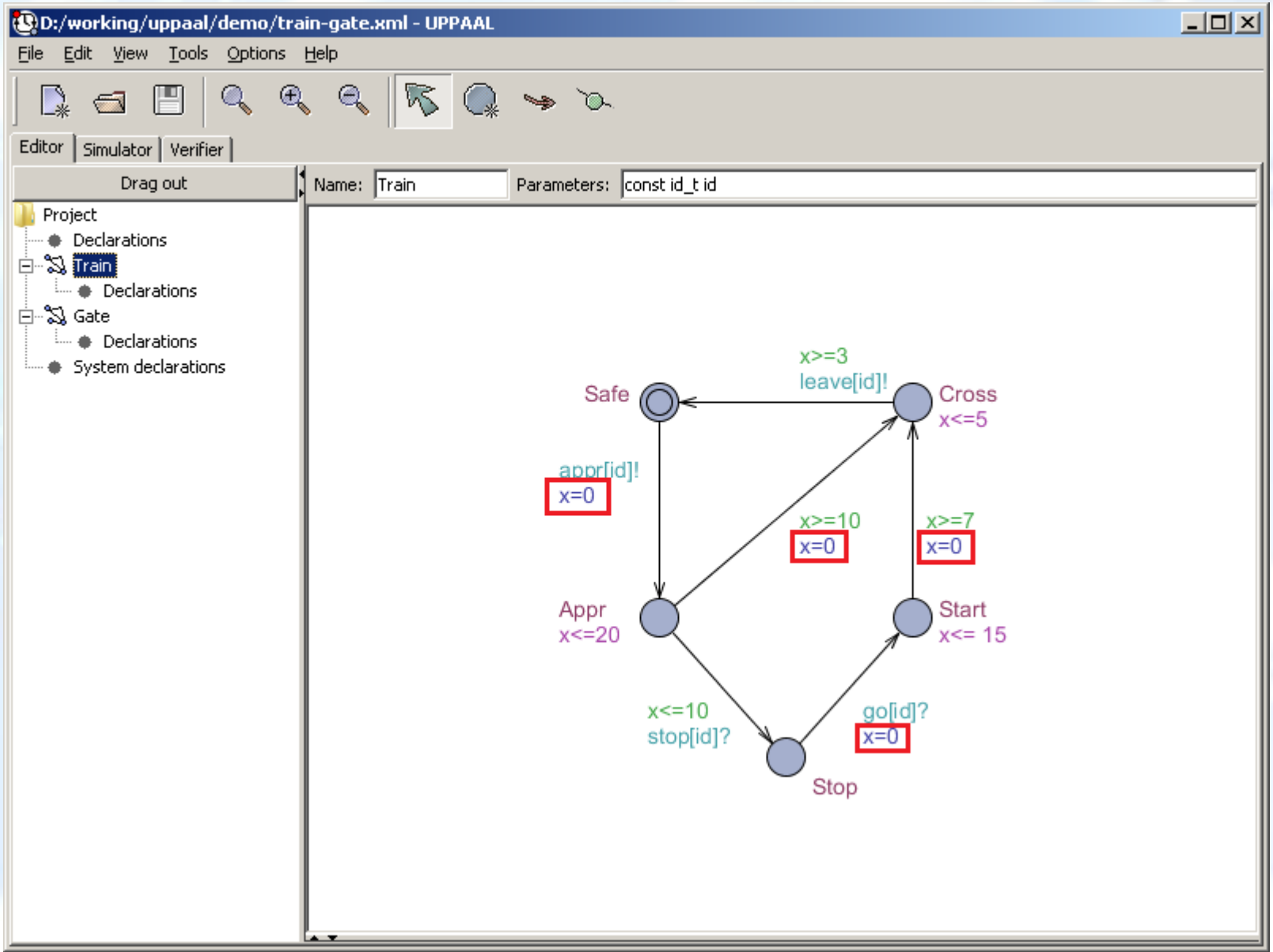
Прием сигналов



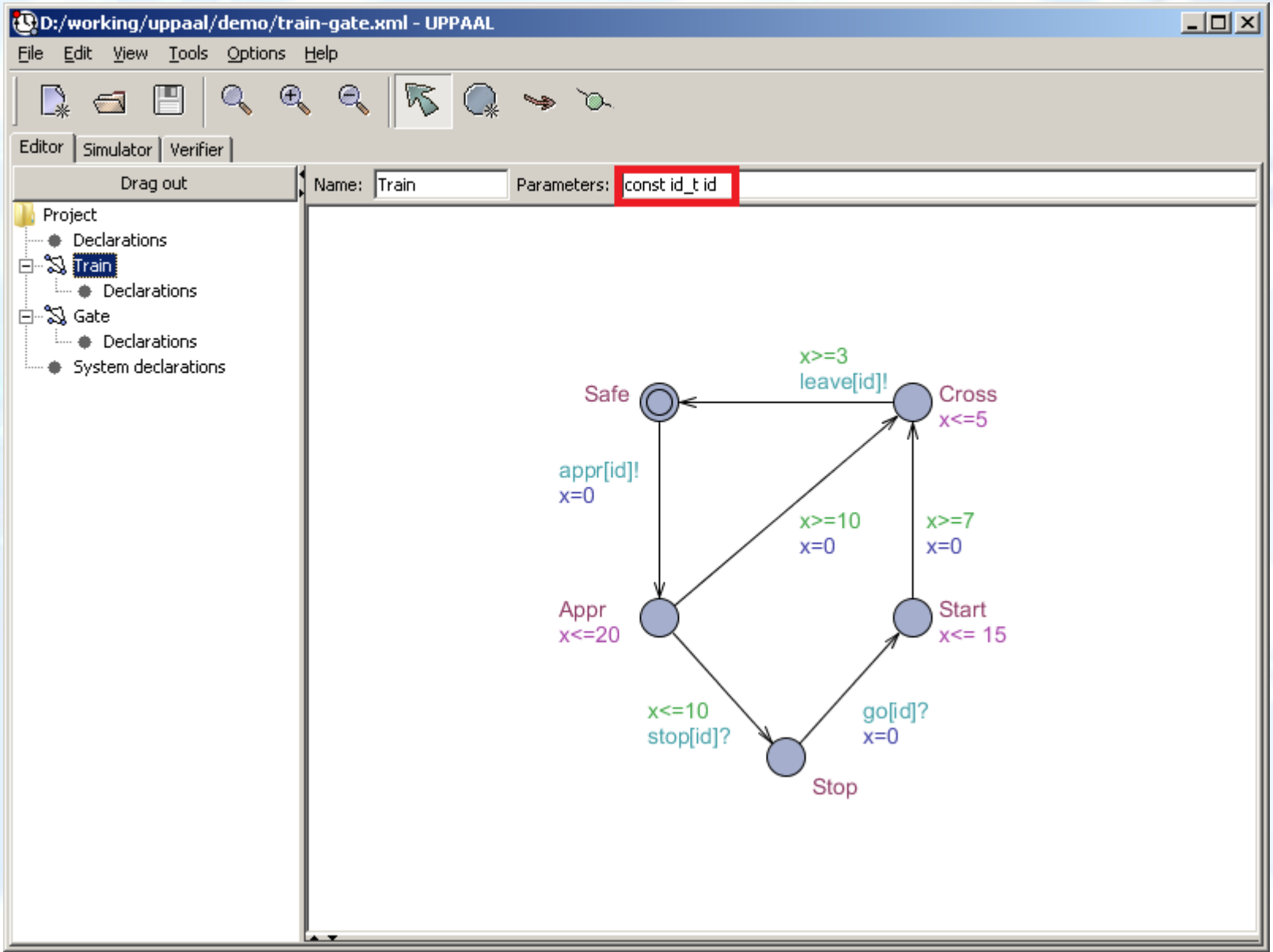
Предусловия



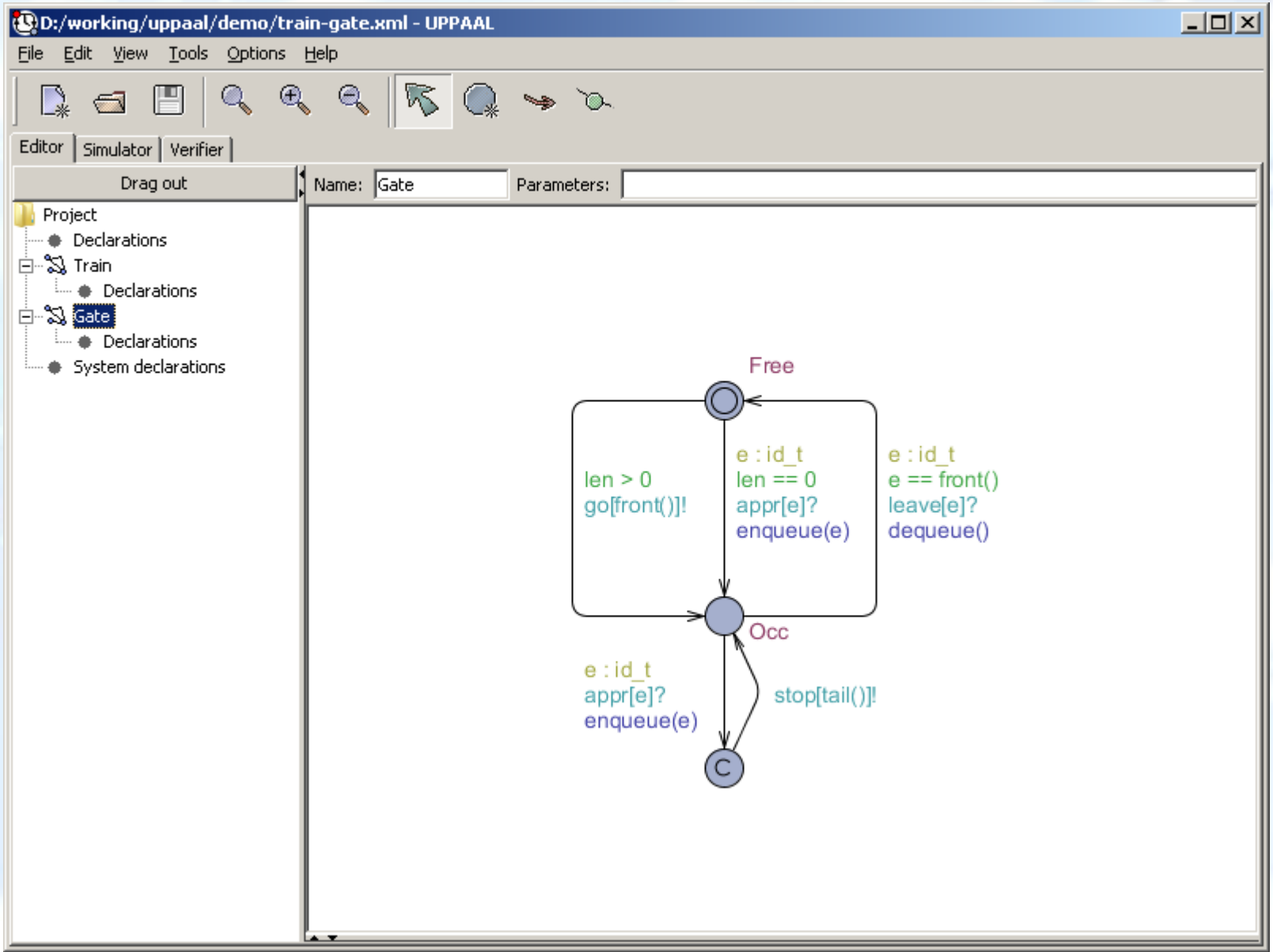
Присваивания



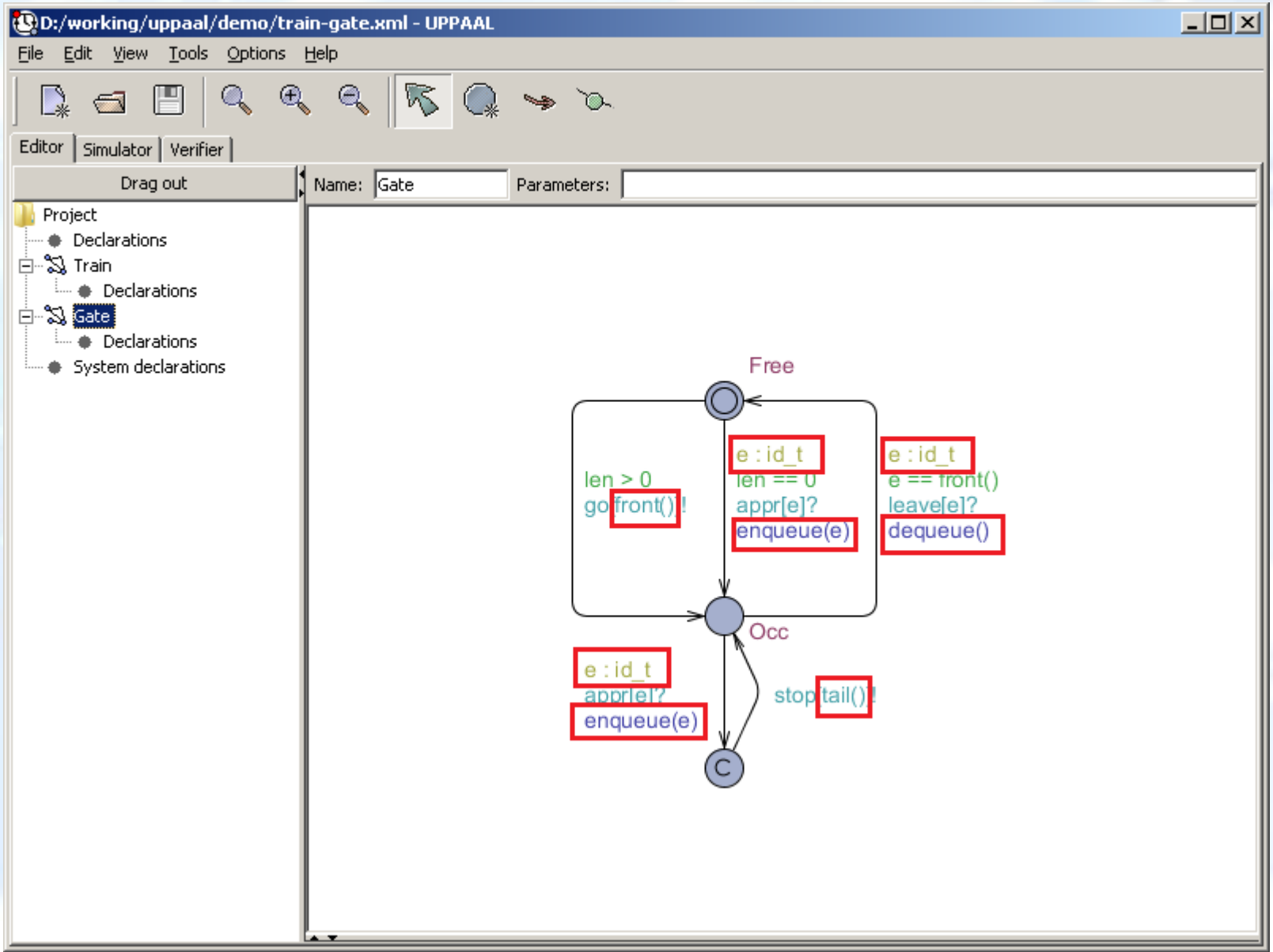
Параметры


D:/working/uppaal/demo/train-gate.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Project
Declarations
Train
Declarations
Gate
Declarations
System declarations
Name: Train
Parameters: const id_t id
x>=3
leave[id]!
Safe
Cross
x<=5
appr[id]!
x=0
x>=10
x=0
x>=7
x=0
Appr
x<=20
Start
x<= 15
x<=10
stop[id]?
go[id]?
x=0
Stop

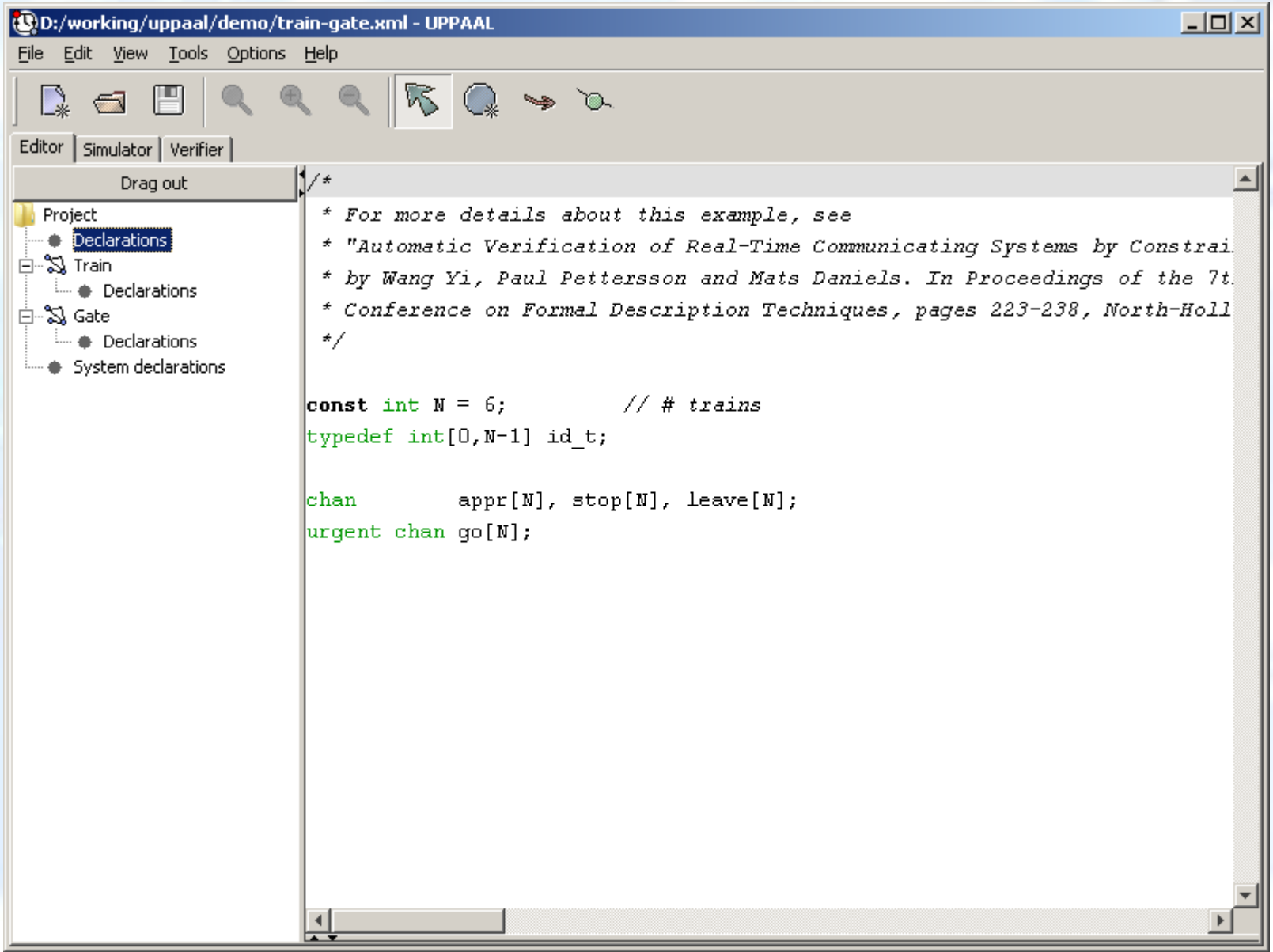
Модуль описания



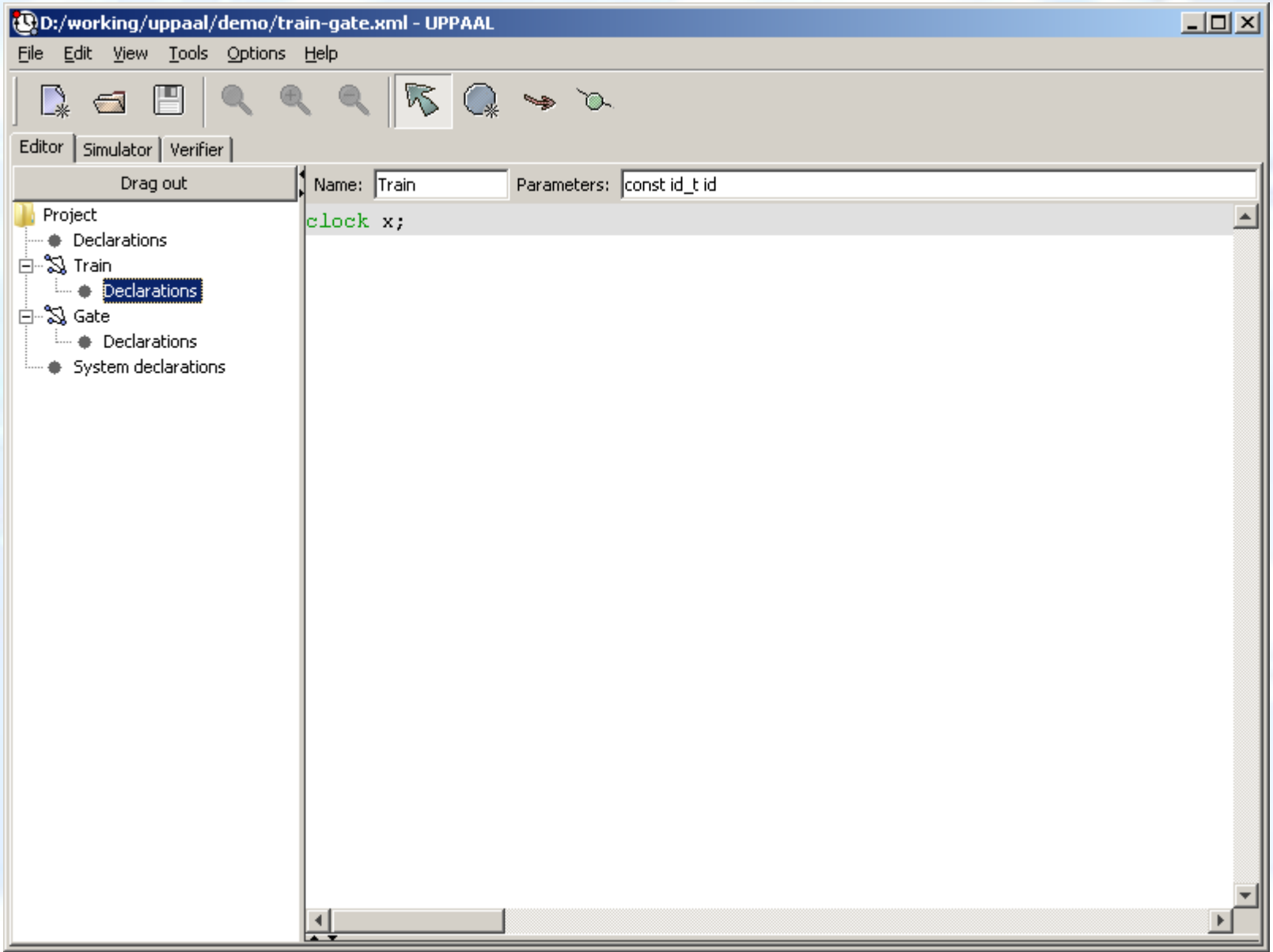
Дополнительные возможности



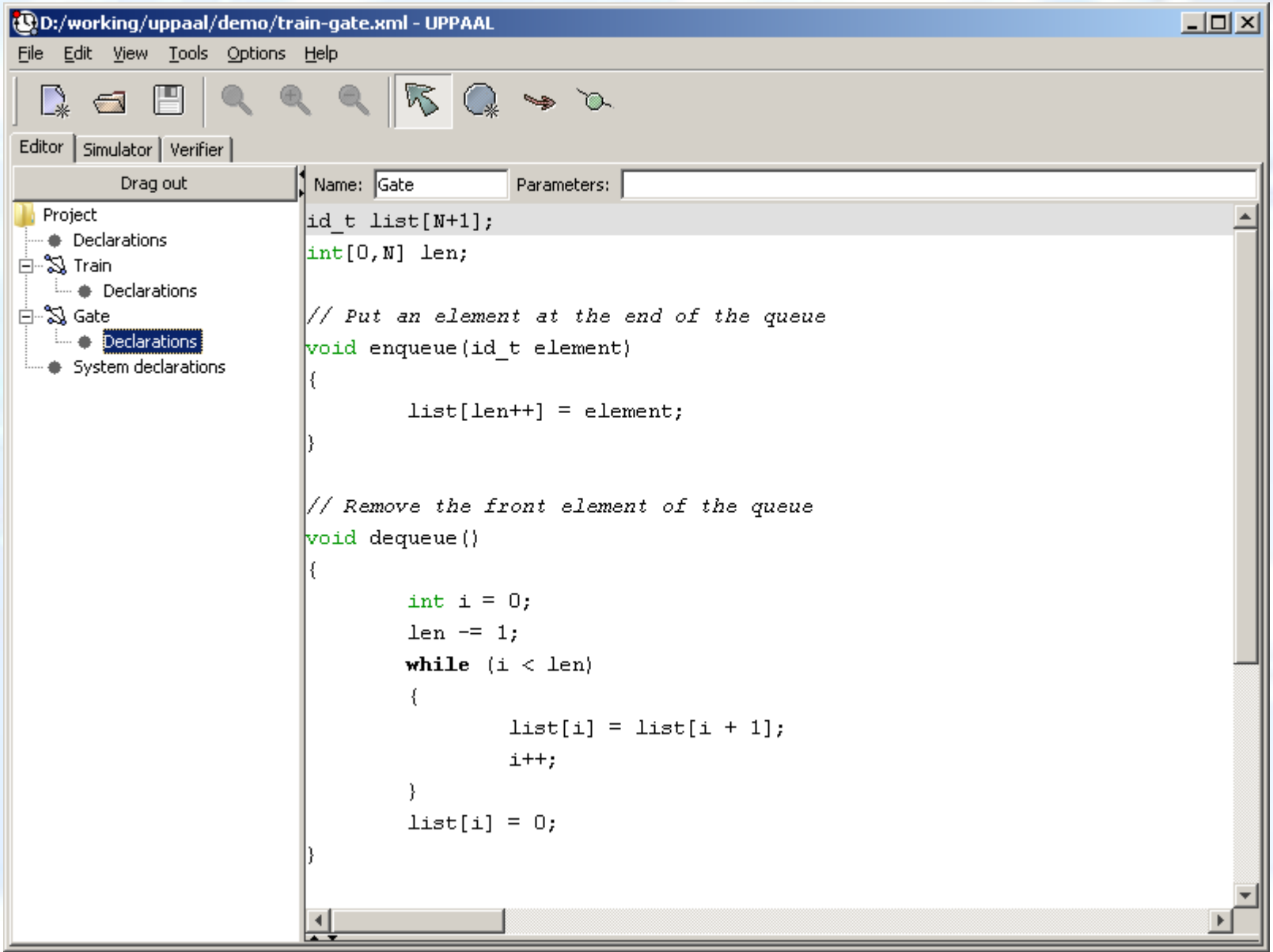
Глобальные описания



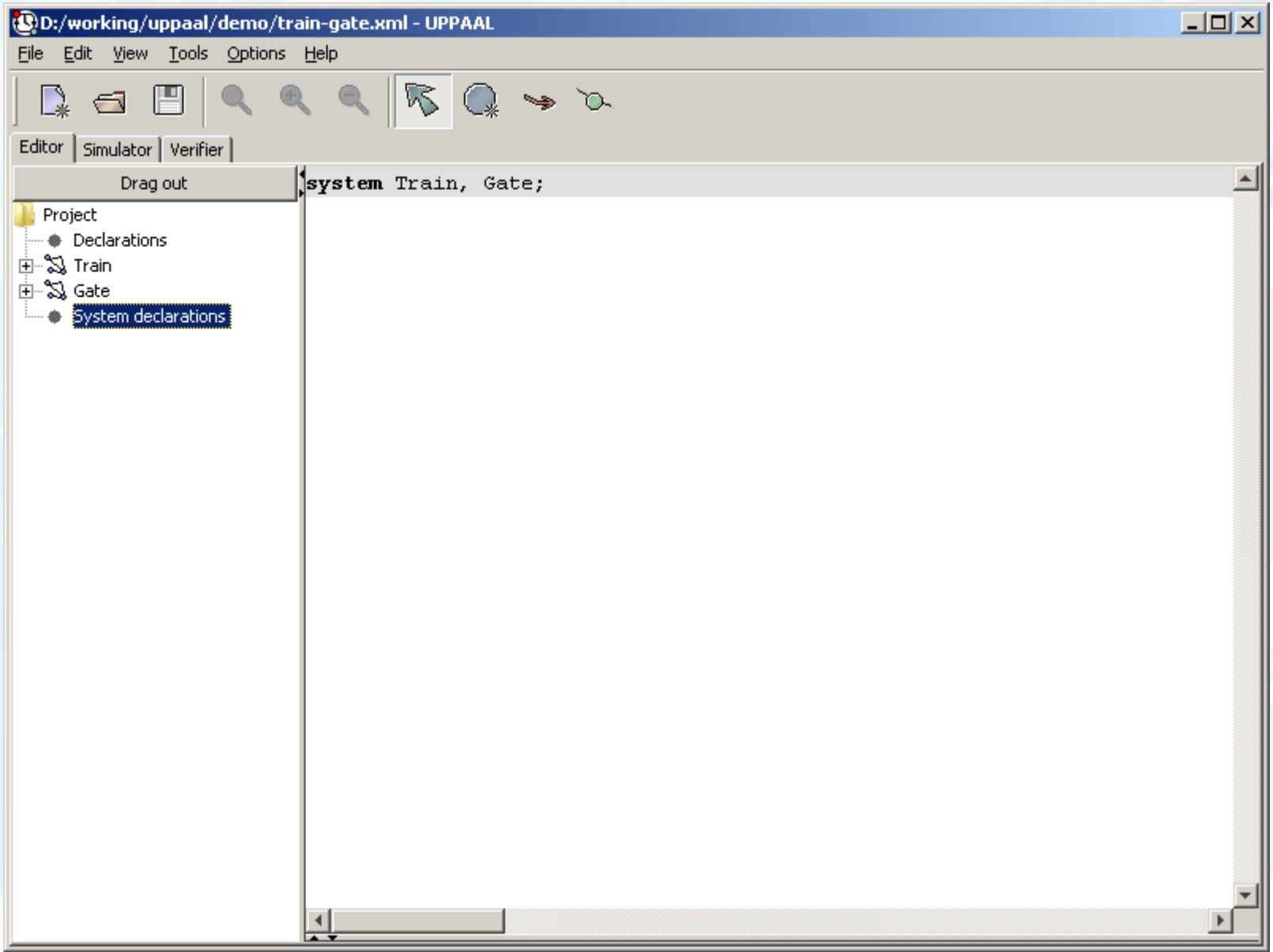
Локальные описания



Локальные описания



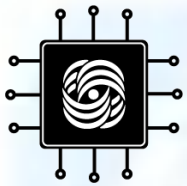
Описание системы



Синтаксис выражений

- Выражение (в стиле языка C)

```
Exp ::= ID | NAT |
    Exp[Exp] | (Exp) |
    Exp++ | ++Exp | Exp-- | --Exp |
    Exp AOp Exp | UOp Exp | Exp BOp Exp |
    Exp?Exp:Exp | Exp.ID
UOp ::= - | ! | not
BOp ::= < | <= | == | != | >= | > | + | - |
    * | / | % | & | | | ^ | << | >> | && |
    || | <? | >? | and | or | imply
AOp ::= := | += | -= | *= | /= | %= | |= |
    &= | ^= | <<= | >>=
```



Синтаксис выражений

Предусловие

- Булево выражение
- Таймеры (t, t′) могут встретиться только в
  `t rel Exp` `t – t′ rel Exp`
- Указанные выше сравнения должны идти в предусловии подряд через конъюнкцию

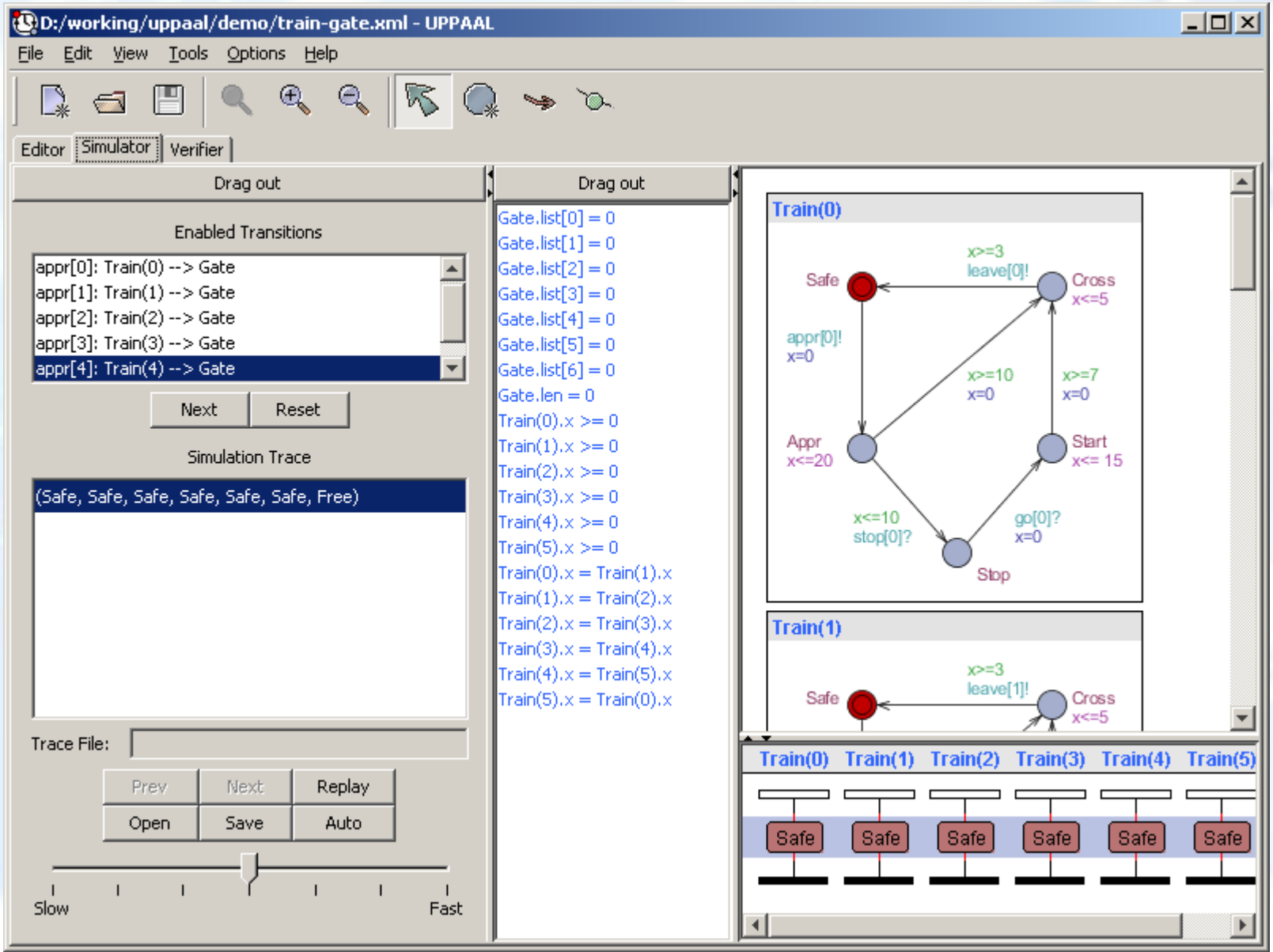
Присваивания

- Разделенные запятой выражения вида
  `x AOp Exp`

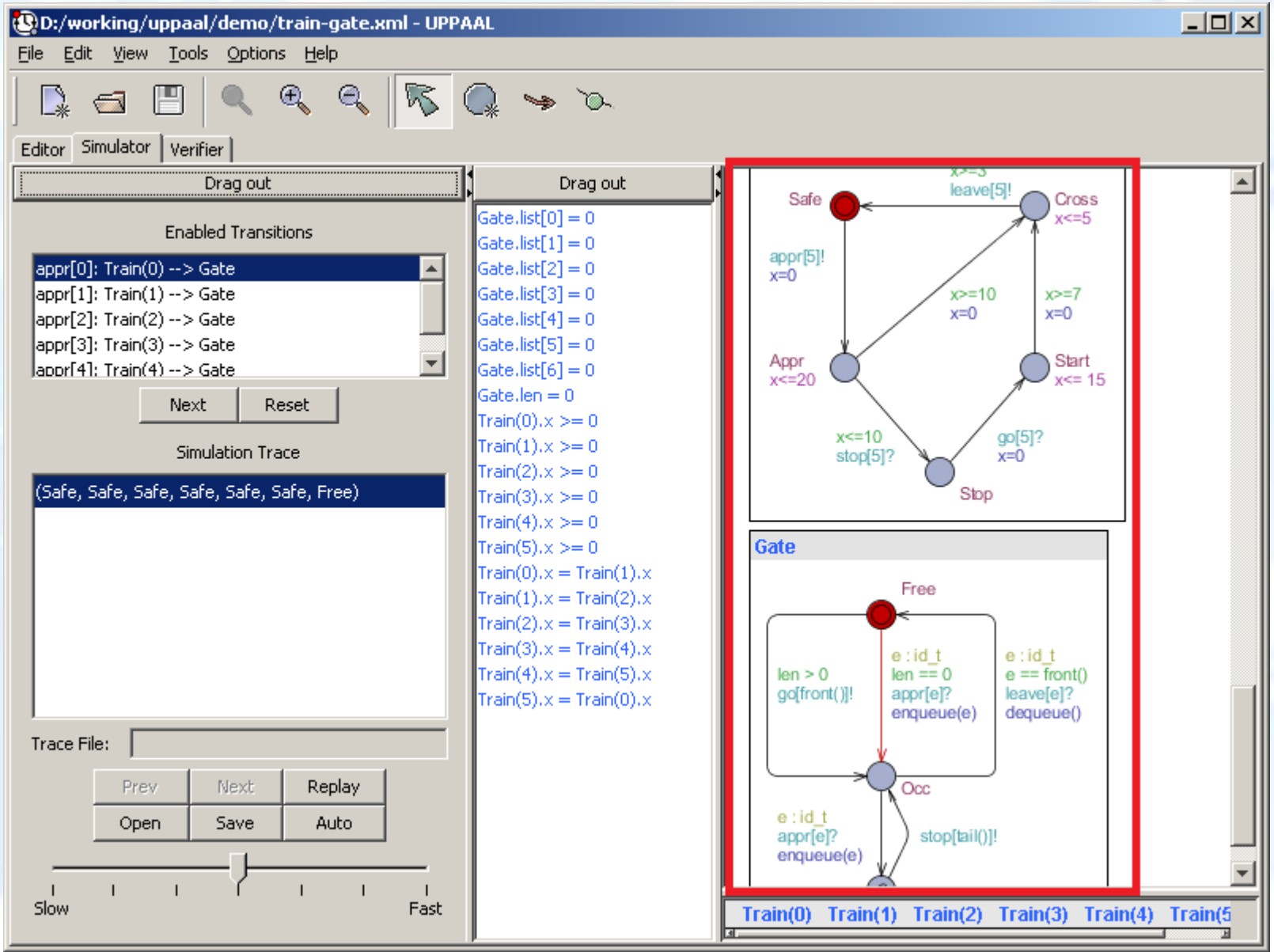
Инвариант

- Булево выражение без отрицаний
- Таймеры (t) могут встретиться только в выражениях вида
  `t rel Exp`

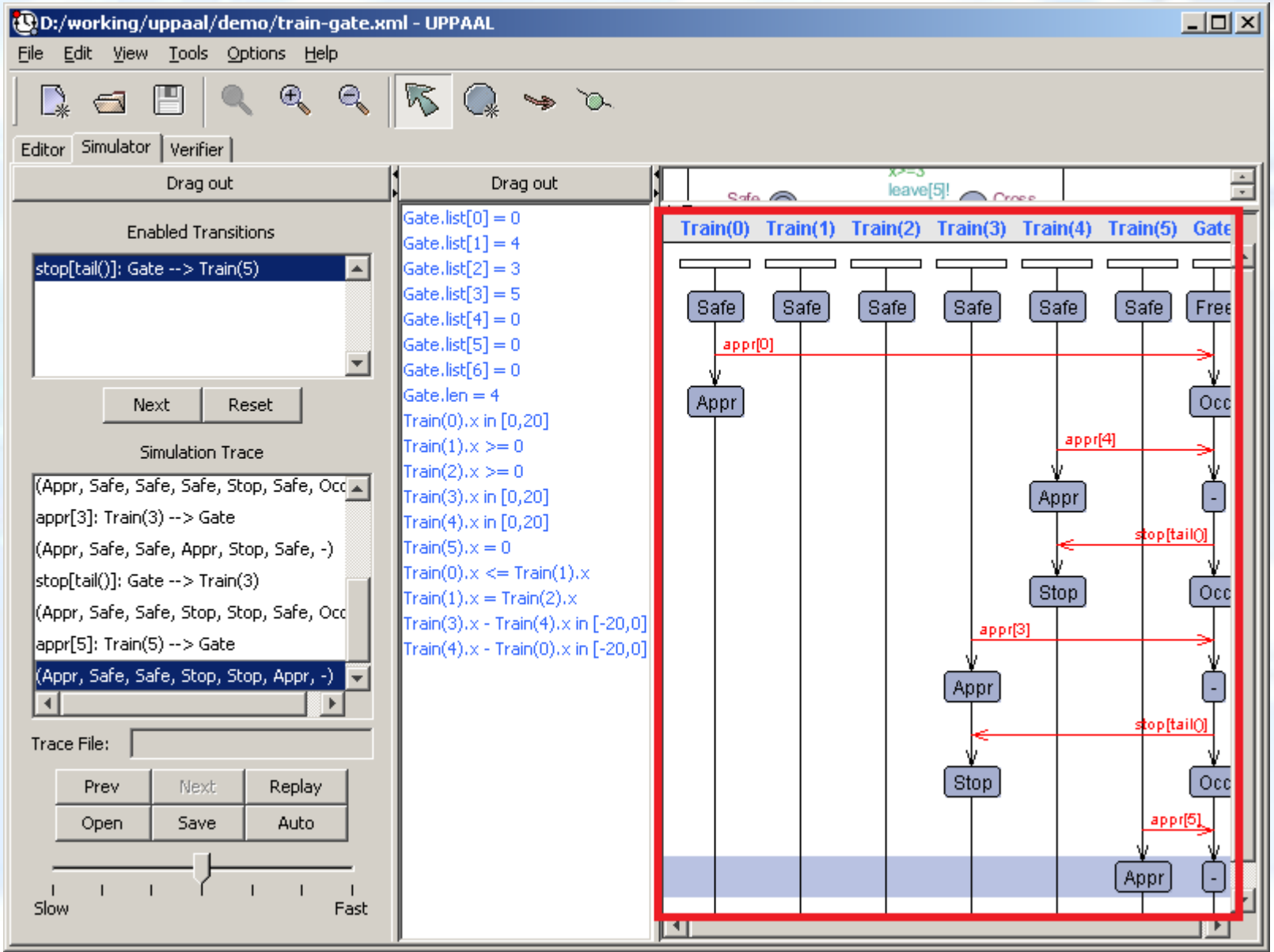
Модуль симуляции



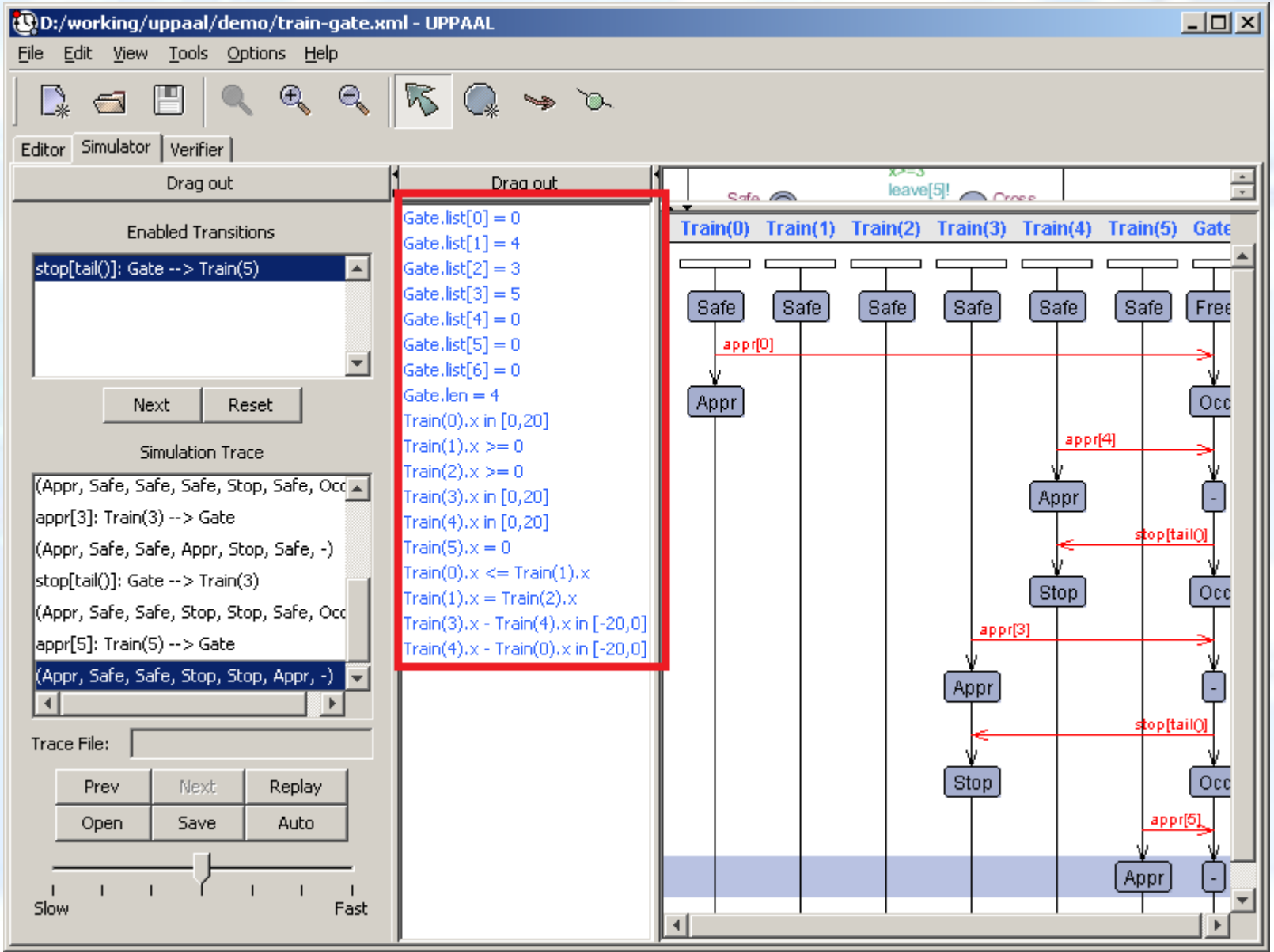
Визуализация активных состояний



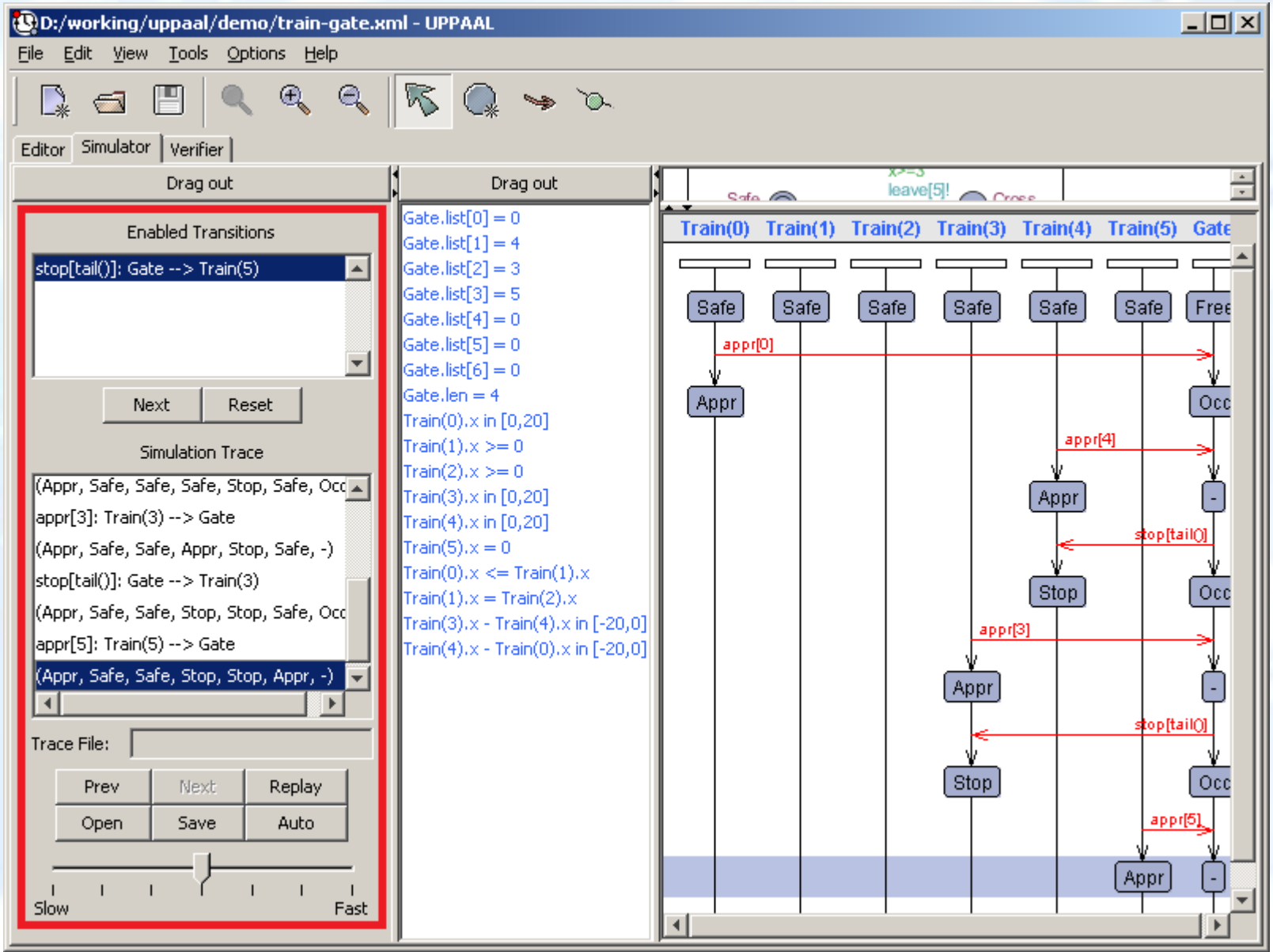
Визуализация трассы



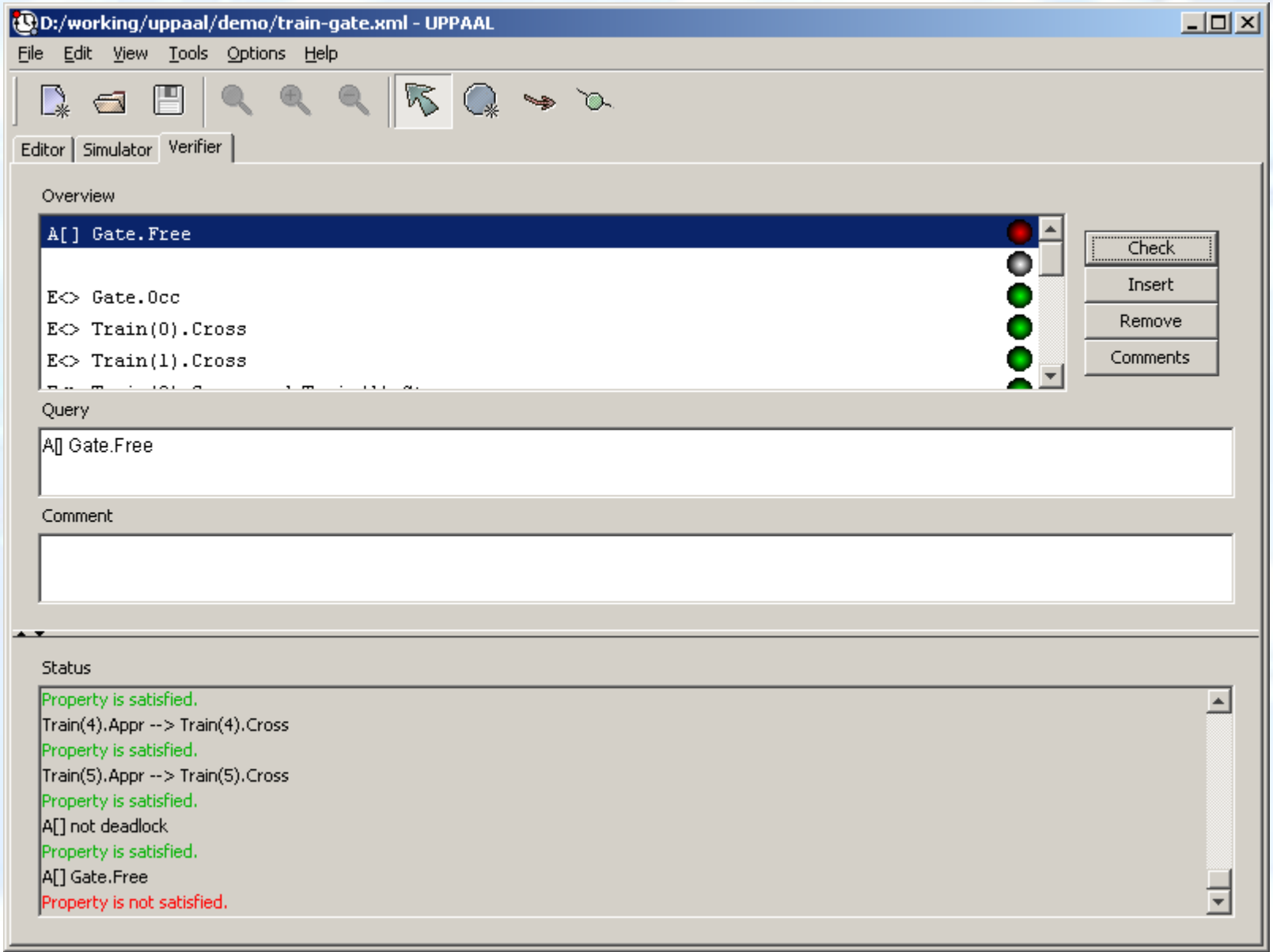
Окно текущих значений данных



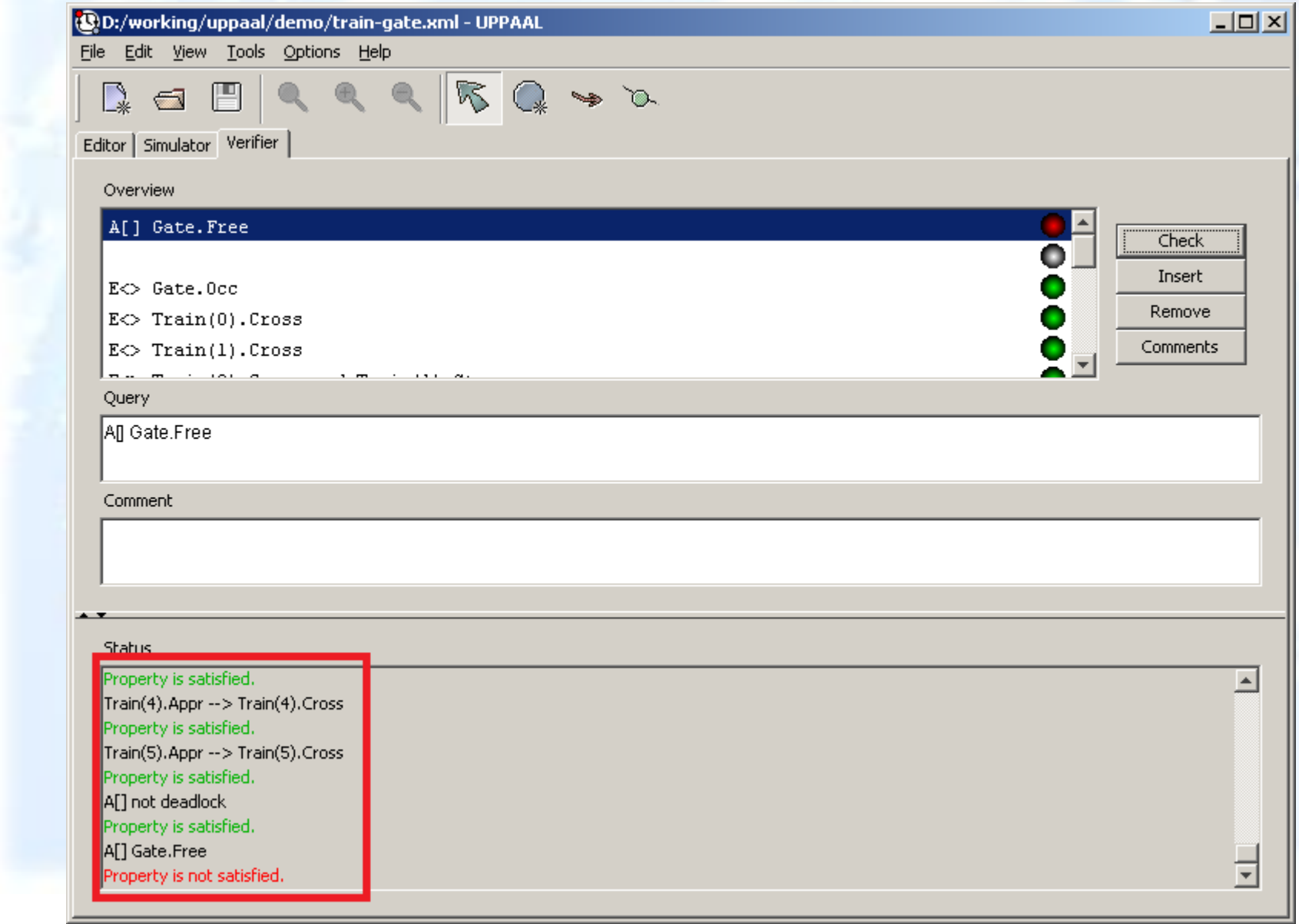
Окно генерации трассы



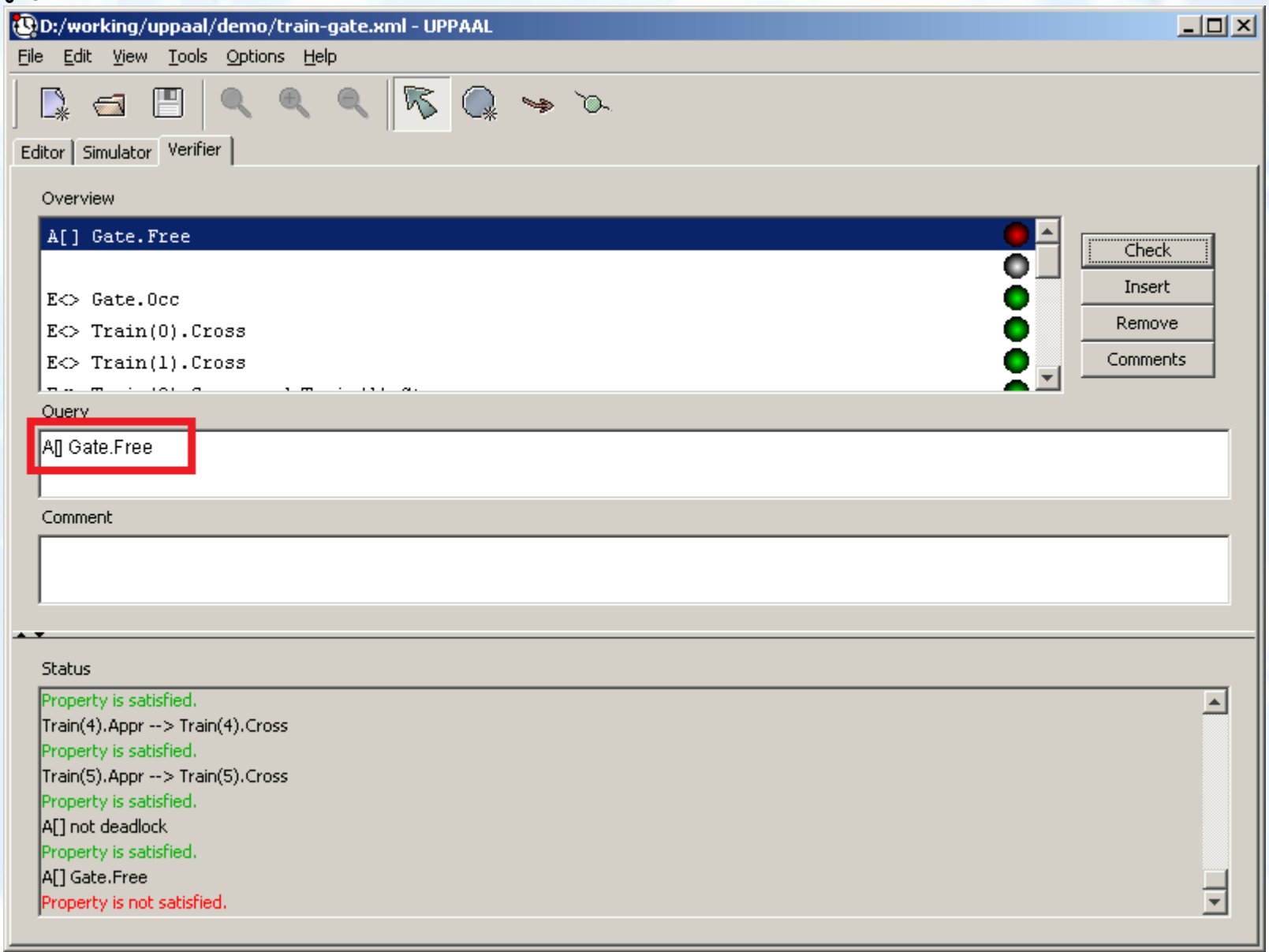
Модуль верификации



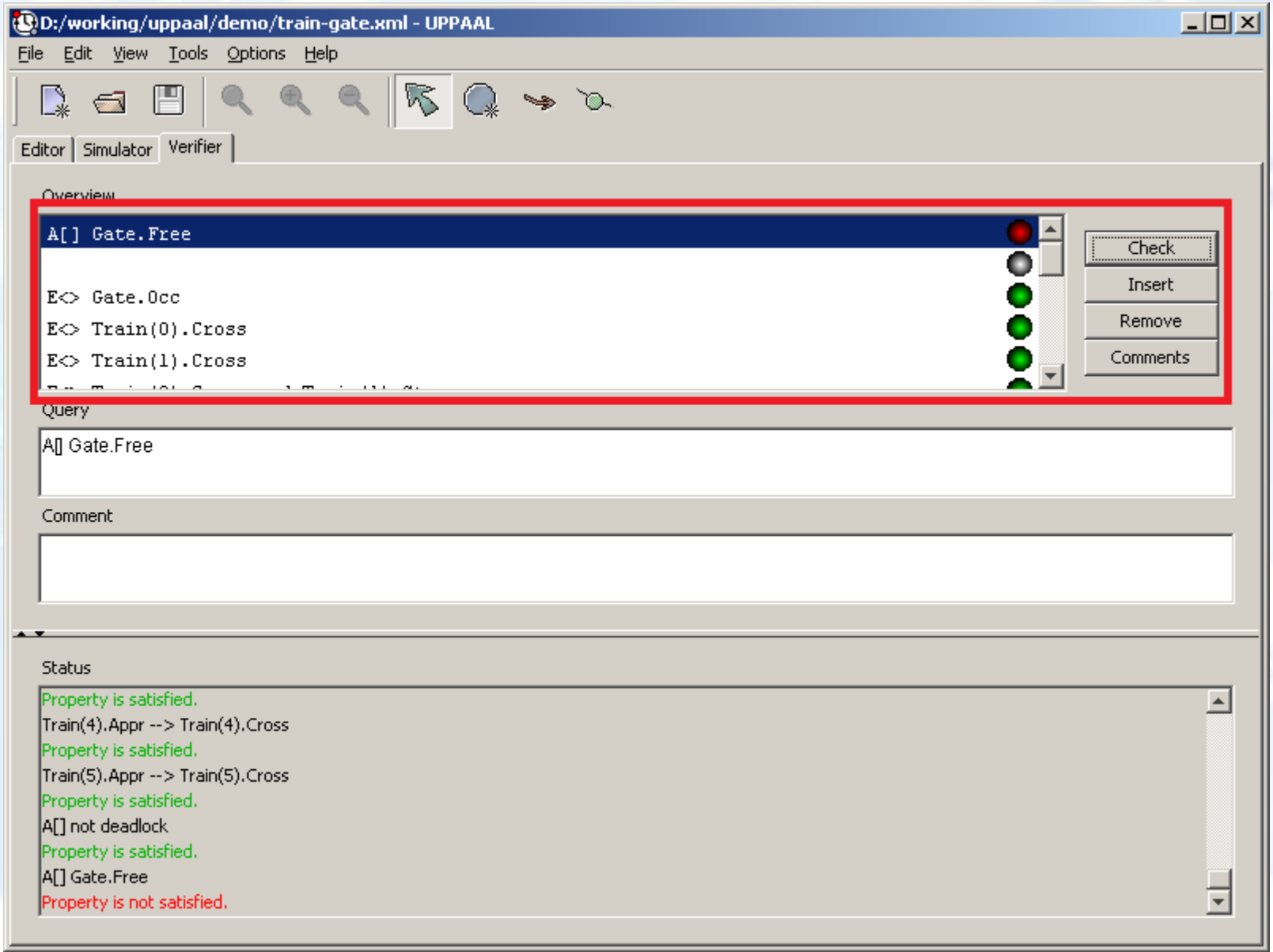
Окно результатов проверки



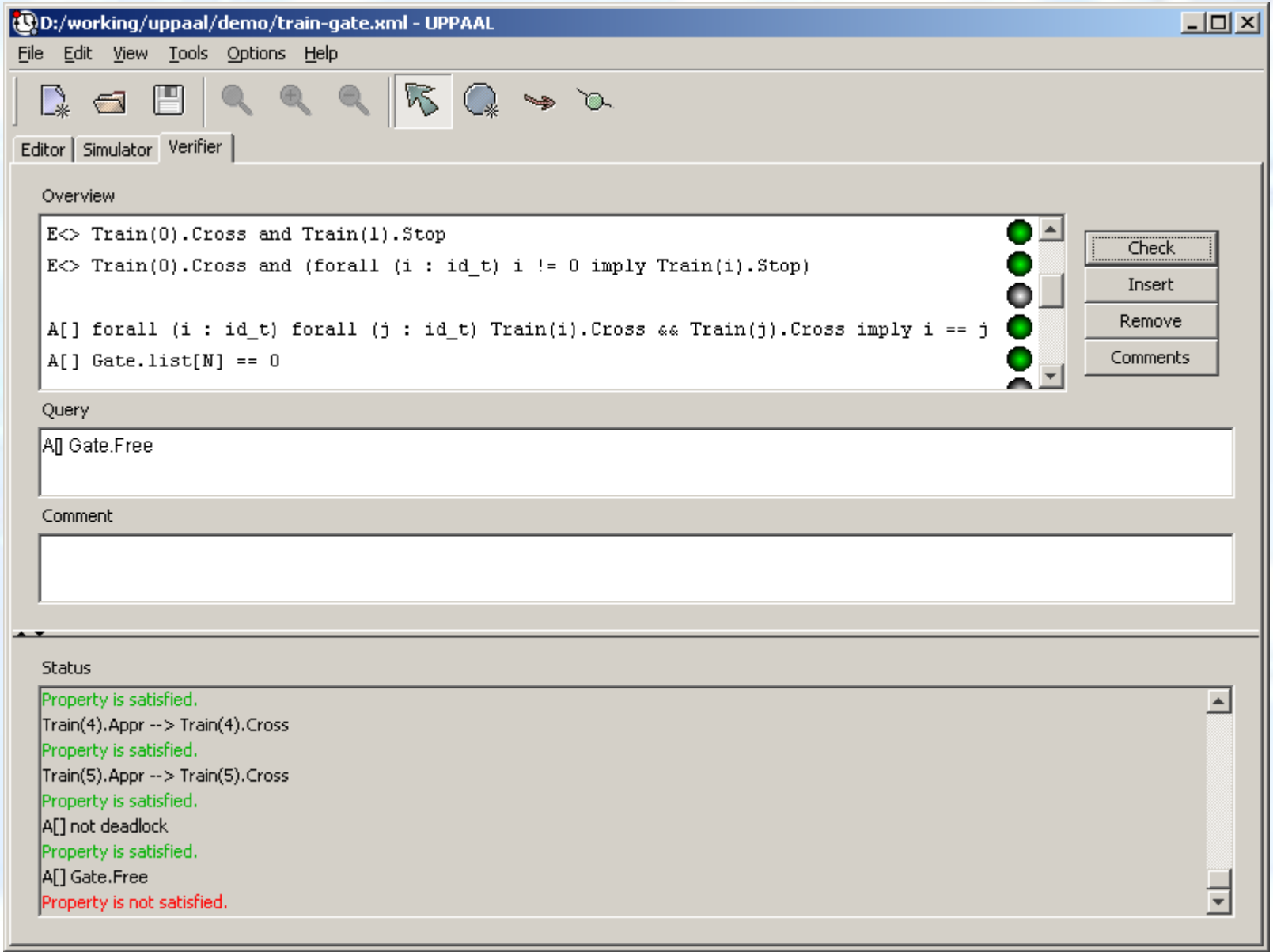
Окно текущего запроса



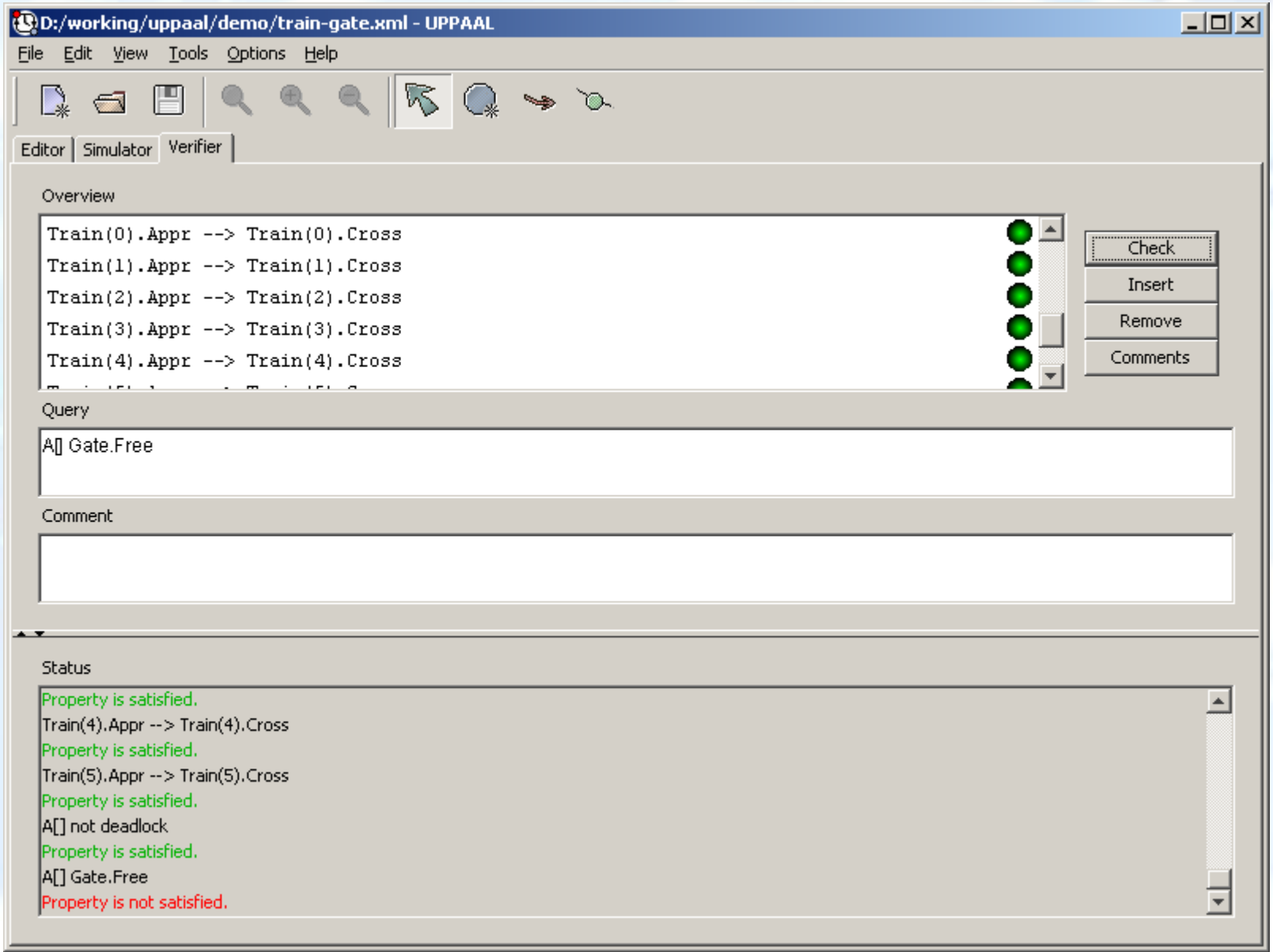
Окно списка запросов



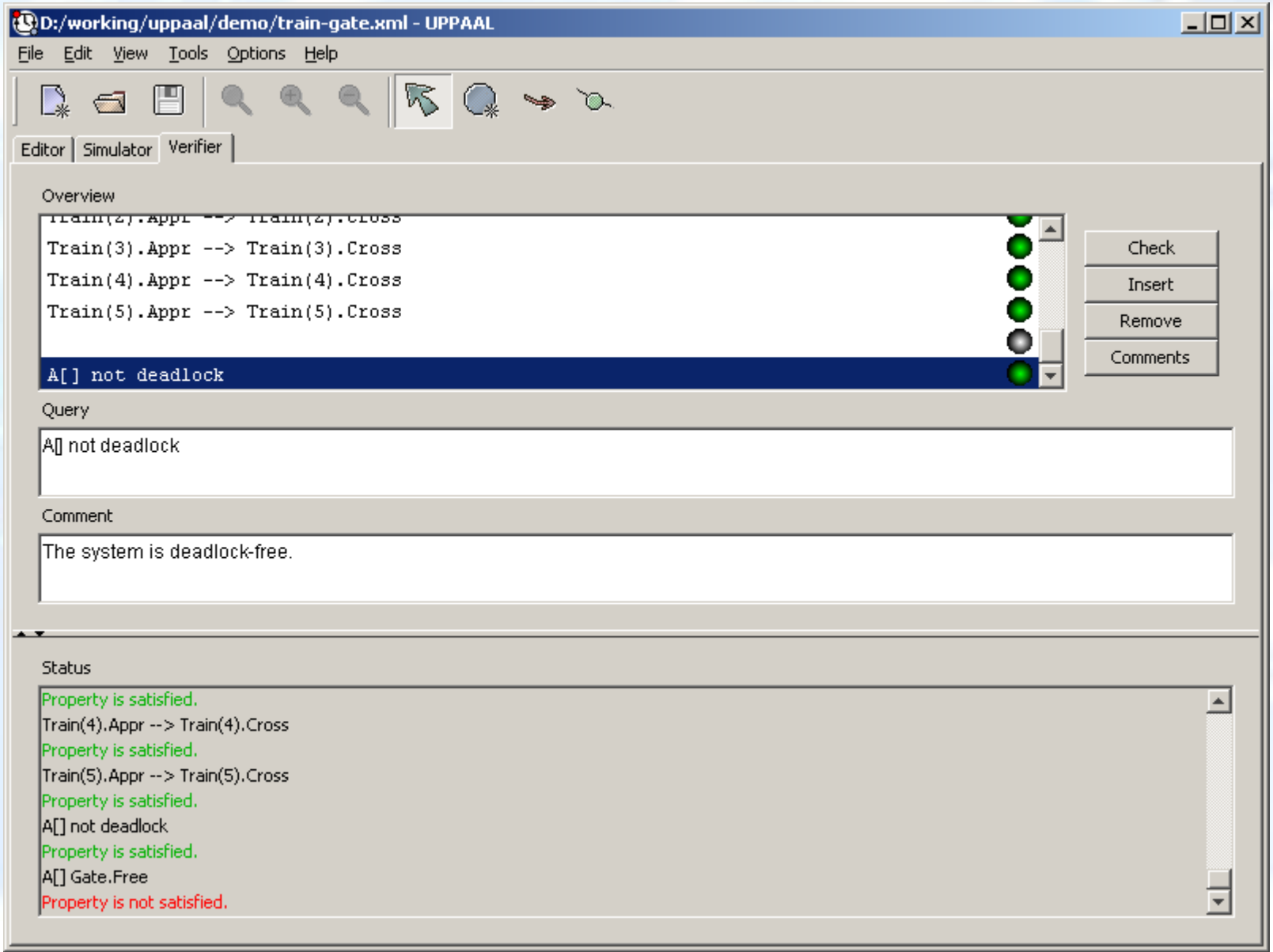
Примеры запросов

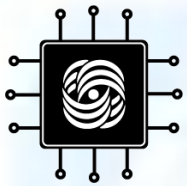


Примеры запросов



Примеры запросов





Язык запросов

- Допустимая формула языка запросов

```
Φ ::= A[] φ | A<> φ | E[] φ | E<> φ
    | φ --> φ
φ ::= Exp | A.l | deadlock
    | φ or φ | φ and φ | not φ | (φ)
```

- deadlock можно использовать только с A[], E<>
- Семантика определяется на основе семантики формул логики CTL
  - A.l ⇔ автомат A находится в состоянии l
  - deadlock ⇔ вычисление после достижения текущей конфигурации невозможно продолжить
  - φ1 --> φ2 ⇔ AG(¬φ1 ∨ AF(φ2))
  - Выполнимость выражений определяется естественным образом

Спасибо за внимание