СИМВОЛЬНАЯ ВЕРИФИКАЦИЯ И ДЕДУКТИВНОЕ ТЕСТИРОВАНИЕ ПРОГРАММНЫХ СИСТЕМ

Александр Летичевский (мл)
Институт кибернетики им. В.М.Глушкова
Национальной Академии Наук Украины
Киев, lit@iss.org.ua
Тел. +380509535646

- **Основные направления работы команды отдела 100 (ранее возглавляемым В.М.Глушковым):**
  - теория автоматов и дискретных преобразователей **;**
  - автоматизация доказательств теорем (70-е г.);
  - параллельное программирование (первая в мире мультипроцессорная ОС с распределенной памятью);
  - системы автоматизации проектирования;
  - системы алгебраического программирования (90-е г.);
  - верификация требований (с 2000 г);
  - система инсерционного моделирования;
  - символьное моделирование формальных спецификаций.

Верификация и тестирование формальных спецификаций в стандартном процессе разработки программного обеспечения

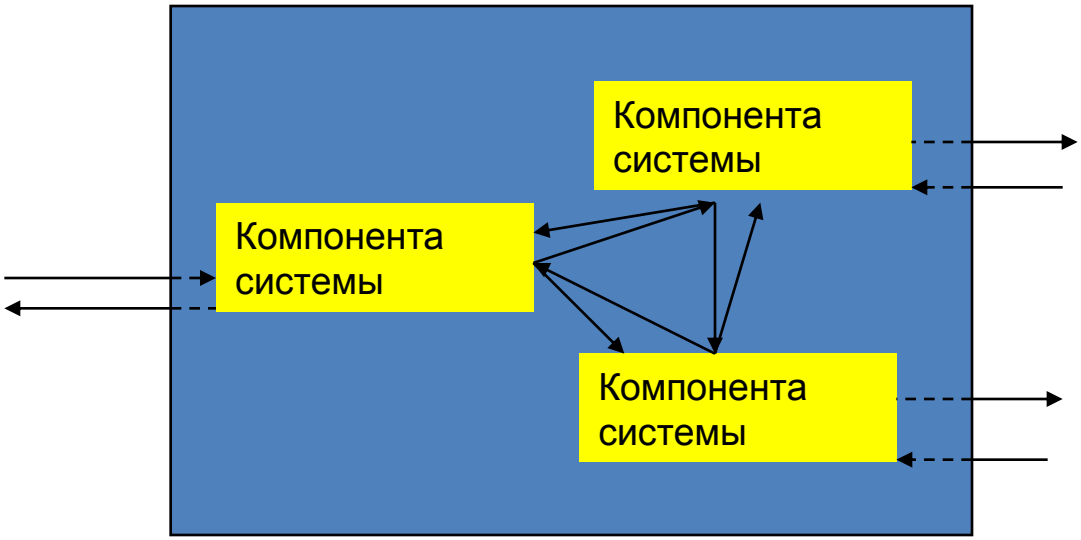
Этап сбора и обработки требований	Этап дизайна	Этап кодирования	Этап тестирования
ТРАДИЦИОННЫЕ ТЕХНОЛОГИИ: формальное представление требований, формальная верификация на основе систем проверки моделей **СИМВОЛЬНАЯ ТЕХНОЛОГИЯ**: символьное моделирование и доказательство свойств формальных требований	**ТРАДИЦИОННЫЕ ТЕХНОЛОГИИ:** Системы верификации UML/SDL спецификаций **СИМВОЛЬНАЯ ТЕХНОЛОГИЯ:** доказательство аннотаций, доказательное и символьное моделирование	**ТРАДИЦИОННЫЕ ТЕХНОЛОГИИ:** Системы автоматического анализа кода СИМВОЛЬНАЯ ТЕХНОЛОГИЯ: доказательство аннотаций	**ТРАДИЦИОННЫЕ ТЕХНОЛОГИИ:** автоматическое создание тестов, автоматизация тестирования **СИМВОЛЬНАЯ ТЕХНОЛОГИЯ:** дедуктивное тестирование

Часть 1
Верификация требований

Какие системы являются объектами верификации?

- Реактивные системы:

1. Внешнее воздействие
2. Состояние среды
3. Изменение состояния среды



В основном требования описывают реакцию системы на внешние воздействия в виде изменения среды.

Какие системы являются объектами верификации?

- **Требования для Event Driven систем.**

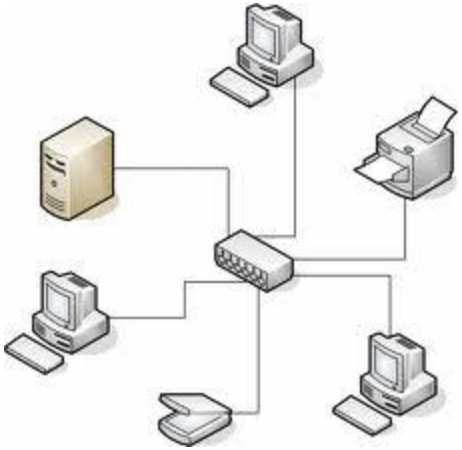
  Описываются как список реакций системы на внешние воздействия

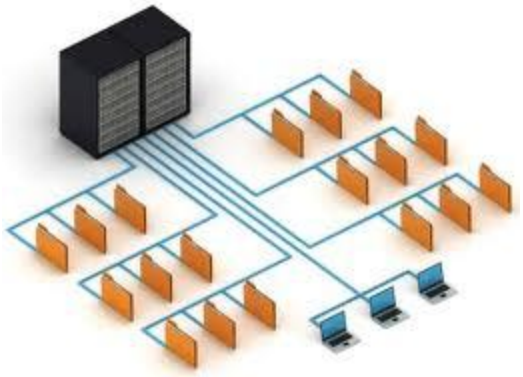
- **Требования для систем с определенным порядком действий (Use Case архитектура).**

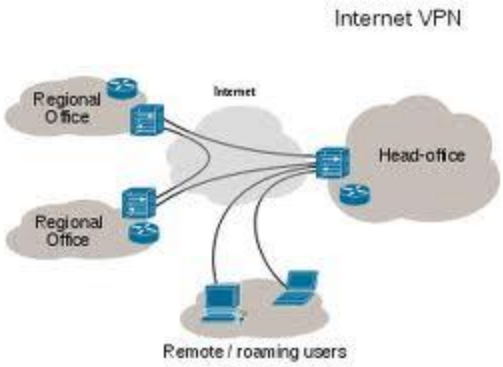
  Описываются как определенная последовательность реакций системы. Такие требования представляются в виде графов, MSC-диаграмм, UCM-диаграмм

- **Требования для синхронных систем**

Распределенные реактивные системы







Индустриальные требования

- Тексты
- Таблицы
- Автоматы
- Диаграммы

8.9 MM Procedures for Successful Inter-CBSC CDMA to CDMA Hard Handoff Execution - Target Side

8.9.1 MM Receives A1: Handoff Request message

This following requirements define the inter-CBSC CDMA hard handoff target execution procedure. This procedure is executed as a result of receiving an A1: Handoff Request message from the MSC. MM validates the handoff request and determines whether the call should be directed towards SDU-SDF or the XC subsystem.

--this should be a parsing req.

CP-HOPC.R.63286 CPSFS-MMHHOTGT-1730 | CI: DNP-MM
4813A, 6088 | Tier: 1 - Top Level
Req: [63286] Upon receipt of an A1: Handoff Request message from the MSC, the MM shall parse the message per section 8.9.23.1, "MM Parses A1: Handoff Request Message".

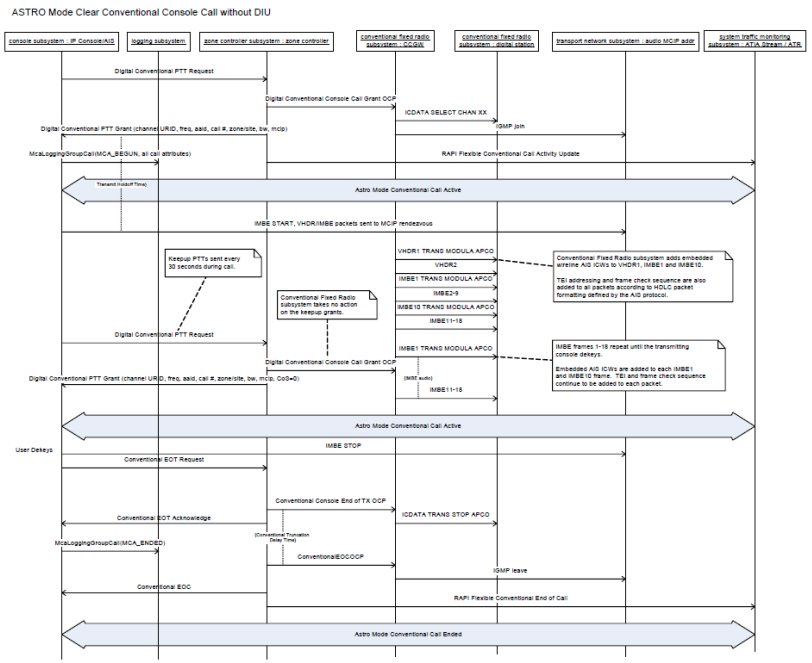
CP-HOPC.R.61934 CPSFS-MMHHOTGT-2 | CI: DNP-MM
6088, 7037 | Tier: 2
Req: [61934] BAND_CLASS shall be retained for the du
BAND_CLASS shall be set to the value of the environmental
it exists, otherwise BAND_CLASS shall be set to the parame
ated with the first CBSC-provisioned cell within the Cell Ide
Request.

SFS-HHOTGTXC-183 SFS-HOPC-C

I: DNP-MM
Tier: 2
ee List of Entries" field within the C
ıdoff Request Message is "1", and no
at matches the target CBSC Band Cla
the handoff per the requirements at

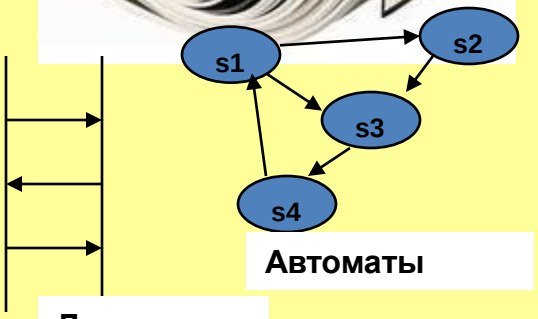
Table 12-3—Applicant state table

EVENT	STATE	VA	AA	QA	LA	VP	AP	QP	VO	AO	QO	LO
	transmitPDU!	sJ[E,I] AA	sJ[E,I] QA	-x-	sLE VO	sJ[E,I] AA	sJ[E,I] QA	-x-	-x-	-x-	-x-	sE VO
	rJoinIn	AA	QA	QA	LA	AP	QP	QP	AO	QO	QO	AO
	rJoinEmpty	VA	VA	VA	VO	VP	VP	VP	VO	VO	VO	VO
	rEmpty	VA	VA	VA	LA	VP	VP	VP	VO	VO	VO	VO
	rLeaveIn	VA	VA	VP	LA	VP	VP	VP	LO	LO	LO	VO
	rLeaveEmpty	VP	VP	VP	VO	VP	VP	VP	LO	LO	LO	VO
	LeaveAll	VP	VP	VP	VO	VP	VP	VP	LO	LO	LO	VO
	ReqJoin	-x-	-x-	-x-	VA	-x-	-x-	-x-	VP	AP	QP	VP
	ReqLeave	LA	LA	LA	-x-	VO	AO	QO	-x-	-x-	-x-	-x-
	Initialize	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO



Этапы верификации требований

Текстовые требования



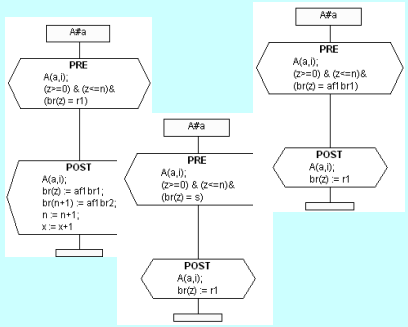
Автоматы

Диаграммы

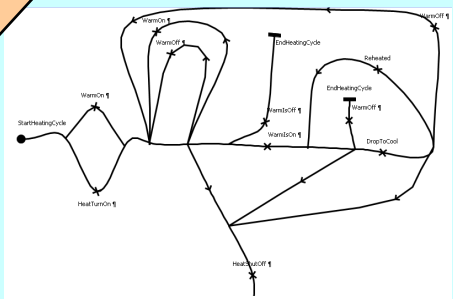
Data1	Data2	Data3
12	14	X
13	15	Y

Таблицы

ФОРМАЛИЗАЦИЯ



Базовые протоколы



UCM-диаграммы

ВЕРИФИКАЦИЯ

Верификационный вердикт

Трассовая генерация

Трассы (тестовый набор)

Генерация модели (кода)

Сгенерированный код

Агенты и типы агентов

- Агент – сущность, которая может быть определена, как объект взаимодействующий с внешним миром. Тип агента определен некоторым множеством атрибутов
- Атрибут - некоторая количественная характеристика агента, выражающая некоторое его свойство.

Button state - “ON”, “OFF”

Temperature of water – t <= 100^{o} C

Position of cover - “OPEN”,”CLOSED”

Color of cover – “BLACK”

Serial number – integer value

GPS coordinates – special values

Триггеры

Нажатие кнопки

Изменение температуры

Получение сигнала

MS DOS

failed

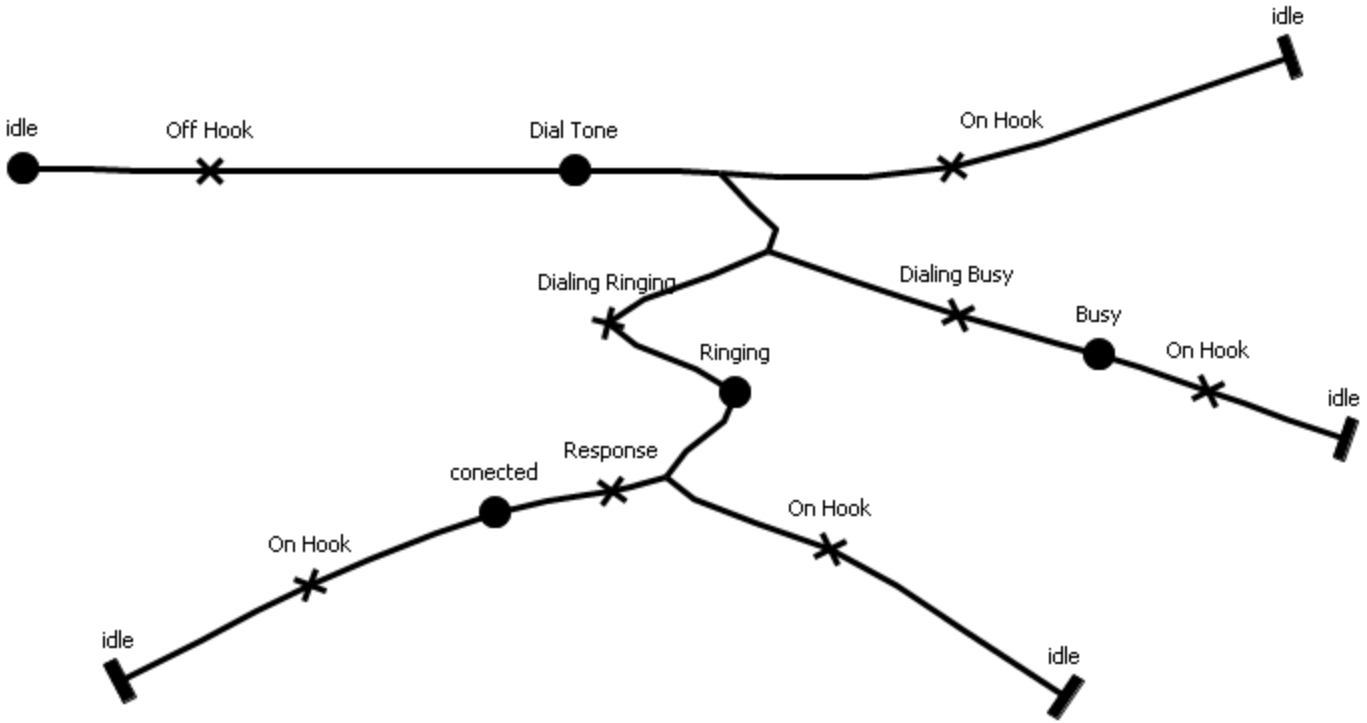
Предупреждение об ошибке

Истечение таймера

Активизация устройства

Триггер – событие воспринимающее реактивной системой

Порядок действий и состояния ожидания



Символьное и конкретное состояние среды

- Конкретное состояние среды определяется конкретными значениями аттрибутов агента или системы агентов

Button_state = ON

Temperature_of_water = 10

Position_of_cover = OPEN

Color_of_cover = BLACK

Serial_number = 1436782

GPS_coordinates_x = 1327436553

GPS_coordinates_y = 8574535729

- Символьное состояние среды определяется множеством конкретных состояний агента или системы агентов в виде формулы над атрибутами агентов.

(Button_state = ON) |/ (Button_state = OFF) & (Temperature_of_water >= 10)&

(Position_of_cover = OPEN) & ~(Color_of_cover = BLACK)&

(Serial_number >1436782) & ~(GPS_coordinates_x > GPS_coordinates_y)

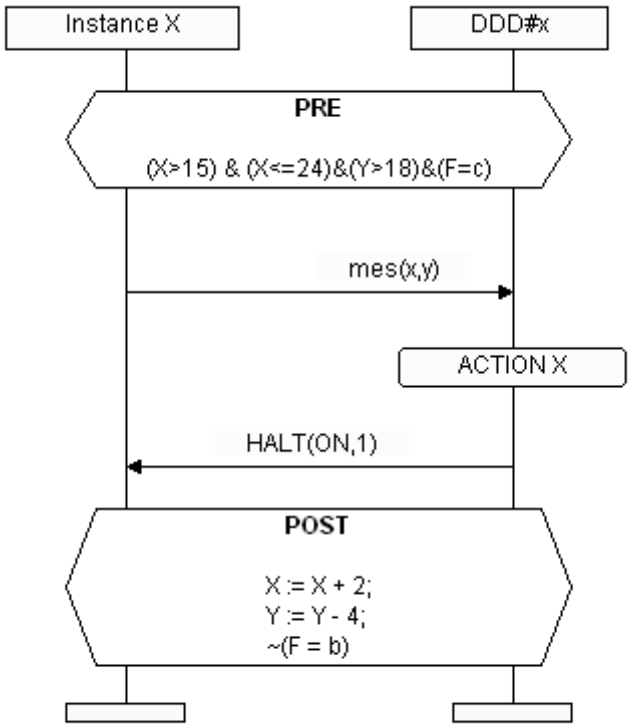
Типы атрибутов

	Операции	Отношения	Пример литерала	Пример изменения среды императивный	Пример изменения среды символьный
Целочисленные и рациональные	+, -, *, / (только линейная арифметика)	>,<,>=,<=,=	(x + y) <= 10*y	x := y + 4	(X>0)|/~(Y=22)
Булевский	&, |/,~	=, ~=	X	X := (a > 0)	~(X & Y & (Z |/ F))
Очереди	get_from_head (tail) add_to_tail (head) remove_from_head(tail)	=, ~=	get_from_head(x)	add_to_tail, remove_from_head	(get_from_head(x) > 0) & ~(get_from_taill(Y) = ON)
Перечислимые (символьные)	-	=, ~=	BUTTON = ON	BUTTON := OFF	~(WATER = BOILED) |/ (WATER = COLD)

Рассматриваются также неинтерпретированные предикаты (функциональные символы), например, F(x, ON) = 100

Язык базовых протоколов

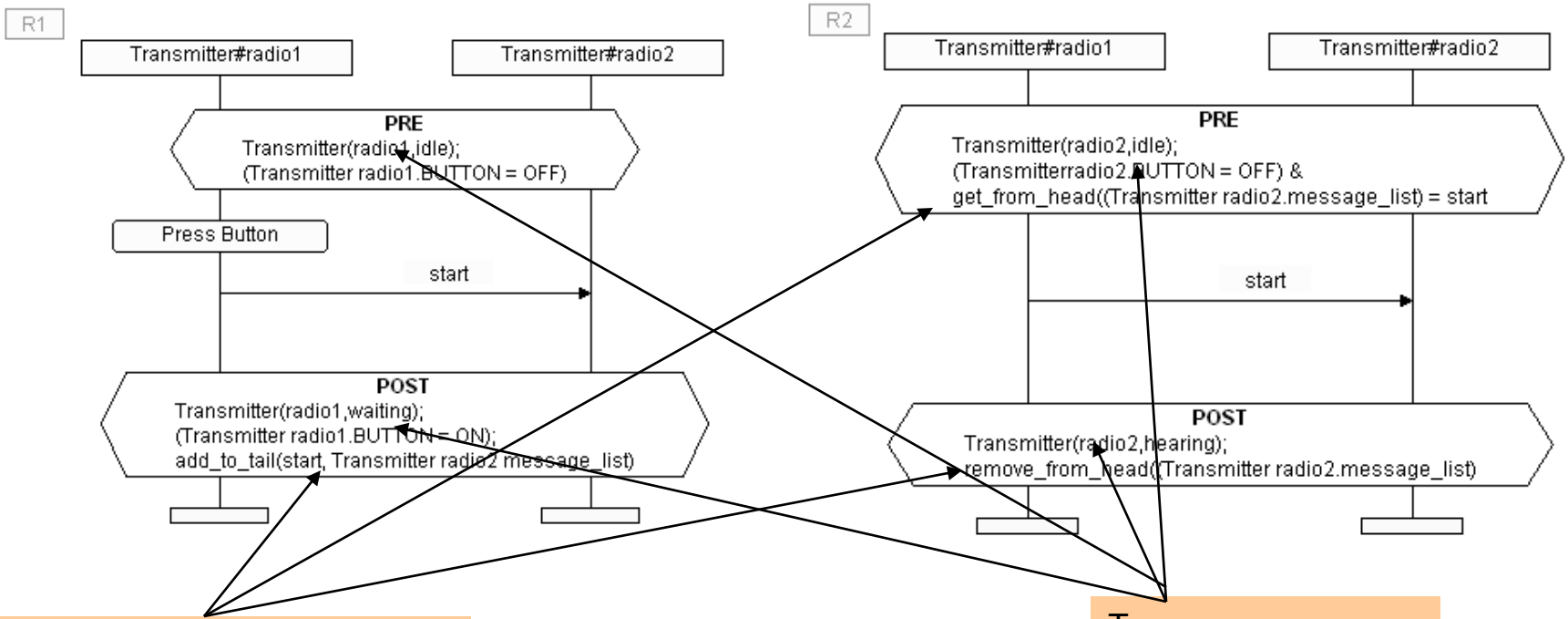
```
mscdocument Protocol10;
msc Protocol10;
DDD#x: instance;
Instance X: instance;
text 'DDDt#x;';
all: condition PRE /*
(X>15) & (X<=24)&(Y>18)&(F=c)*/;
Instance X: out mes(x,y), 1 () to DDD#x;
DDD#x: in mes(x,y), 1 () from Instance X;
DDD#x: action 'ACTION X';
DDD#x: out HALT(ON,1), 2 () to Instance X;
Instance X: in HALT(ON,1), 2 () from DDD#x;
all: condition POST /*
X := X + 2;
Y := Y - 4;
~(F = b)*/;
DDD#x: endinstance;
Instance X: endinstance;
endmsc;
```



ТЕКСТОВОЕ ПРЕДСТАВЛЕНИЕ

ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ

Пример базовых протоколов



Операции очередей, обеспечивающие правильный порядок обработки сообщений

Техника макросов (скрытое присваивание)

Пример требования к реактивной системе

4.1.1.2 Triggering Event

The triggering event for this procedure is different based on the situation. The following two situations are described below:

Initial Network Entry Situation - The MSS has no AK context in which to sign EAP-Start and EAP-Transfer messages with.

- MSS completes ranging and basic capabilities exchange with the BTS.
- MSS has indicated that it is capable of performing EAP authentication.
- BTS triggers EAP Authentication by sending an M_SEC_INIT_AUTH message to the CAPC. The BTS should do this only when the SBC_REQ received from the MSS for capabilities negotiation does not contain the SIQ TLV.

Re-authentication Situation - The MSS has a valid AK context but the MSS's existing context is nearing expiration and the MSS initiates re-authentication to maintain connectivity to the system.

- MSS Authorization Grace Timer has expired so the MSS is going to initiate re-authentication.
- Upon the BTS receiving the EAP Start PKM message from the MSS, the BTS triggers EAP Authentication by sending an M_SEC_INIT_AUTH message to the CAPC.

Предусловие

Триггер

Изменение среды, сопровождающееся некоторым переходом или последовательностью действий

1. Initial Network Entry Situation:

1a) After the BTS receives the SBC-REQ and has sent in response the SBC-RSP, the BTS sends a M_SEC_INIT_AUTH message to the CAPC. The M_SEC_INIT_AUTH CMAC CmacPresent flag is set to 0x00 and the CmacVersion flag set to the CMAC mode negotiated between the BTS and the MSS during Basic Capabilities Negotiation.

Символьное моделирование требований

- Зафиксировав некоторое начальное символьное состояние среды $E_0(X)$, где X – множество атрибутов, мы можем моделировать требования и получать сценарии в виде трасс:

$$E0 \overset{B1}{\rightarrow} E1 \overset{B2}{\rightarrow} E_2 \rightarrow \ldots \rightarrow E_n \overset{Bn}{\rightarrow}$$

- Протокол $B_i = (P_i, W_i, Q_i)$ **применим** если формула $E_{i-1}(X)\&P_i$ выполнима.
- Если протокол применим, то может быть вычислено новое символьное состояние среды с помощью предикатного трансформера *Pt*:

$$Env_i = Pt(Env_{i-1}(X)\&P_i, Q_i)$$

***Литература: Летичевский А.,Годлевский А., Летичевский А. (мл). «Свойства предикатного трансформера» // Кибернетика и Системный Анализ*, 4, 3-16, 2010.**

Инструментарий

- Символьный трассовый генератор

Вход: множество базовых протоколов

Выход: вердикт трассы

Назначение инструмента – определение достижимости свойства

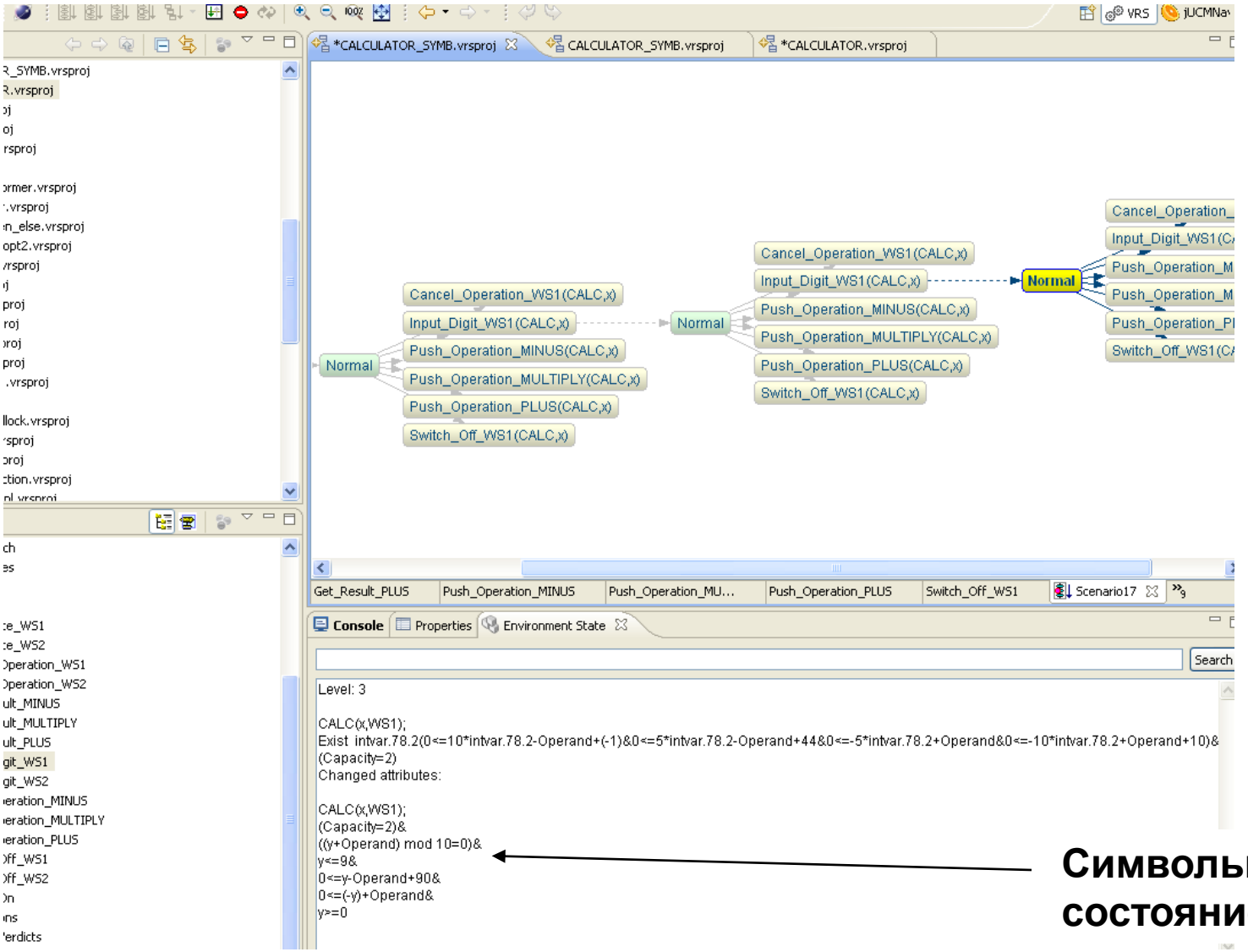
При символьном моделировании генерируется последовательность состояний среды: $E_0(X),E_1(X),\ldots,E_n(X)$

Свойство $W(X)$ достижимо если $E_i(X)\&W(X)$ выполнимо

Доказательные машины

- При определении выполнимости формулы используются следующие базовые средства:
  - решатель для целых чисел, основанный на алгоритме Прессбургера;
  - решатель для рациональных чисел, основанный на алгоритме Фурье-Моцкина;
  - доказательная машина для символьных и перечислимых типов данных;
  - доказательная машина для логики первого порядка

Символьная трассовая генерация



Символьное состояние среды

Отношение следования

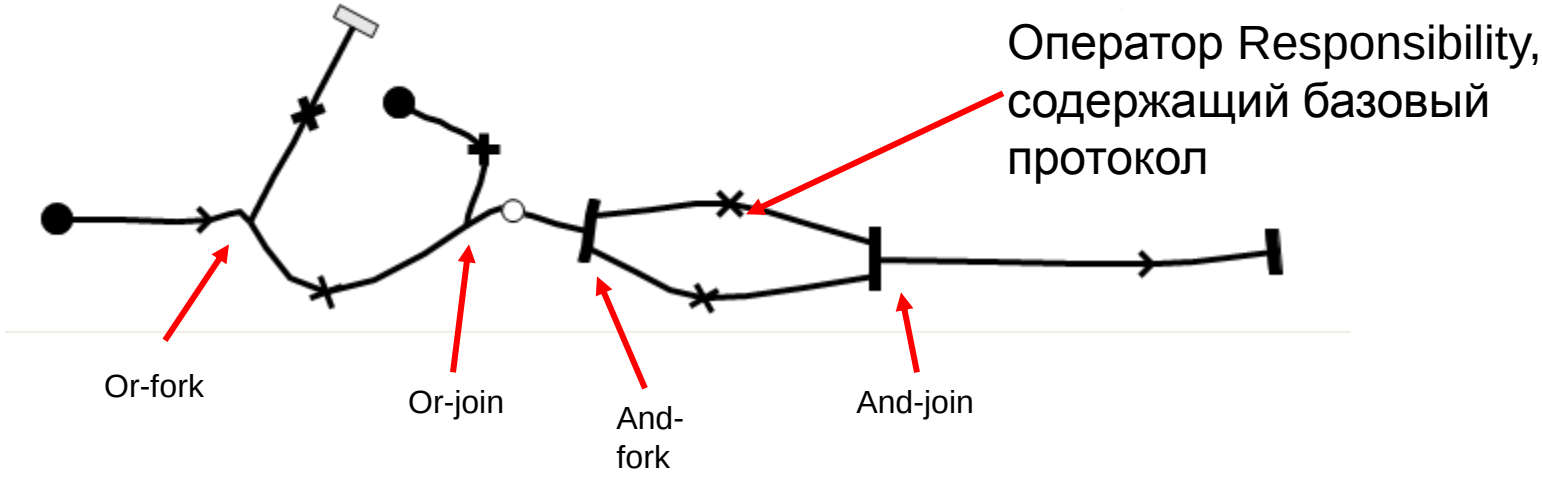
- Множество базовых протоколов формирует отношение следования.
- Сильнейшее отношение следования определяется существованием трассы в которой протокол B_j следует за $B_{i.}$ Обозначим это отношение $Suc\ (B_i,B_j)$
- Отношение следования определяется применимостью протокола или выполнимостью формулы. Если протокол B_j выполним после B_i т.е. $E_i\ \&\ P_j$ то он находится с ним в отношении следования $Suc\ (B_i,B_j)$

E_i – символьное состояние среды после применения базового протокола B_i

P_j – предусловие базового протокола $B_{j.}$

Усилить наислабейшее отношение следование (каждый с каждым) можно с помощью определения выполнимости формулы $E\ \&\ P_j$, где в качестве E можно рассмотреть $Pt(P_i,Q_i)$, Q_i – постусловие протокола B_i

UCM-диаграммы



Для требований с определенным порядком мы рассматриваем формализацию в виде UCM-диаграмм, связывающую базовые протоколы.

UCM диаграмма формирует отношение следования, которое может быть усилено для каждого протокола вычислением E_{i-1} & P_i, где E_{i-1} – состояние среды перед выполнением протокола B_i с предусловием P_i.

Состояние среды может вычисляться статически, как инвариант. Слабейший инвариант вычисляется следующим образом:

$E_i = Pt(E_{i-1}\&P_i, Q_i)$

Статическая проверка требований

- **Непротиворечивость требований.** Непротиворечивая система должна быть детерминирована. Для каждой пары базовых протоколов B_i и B_j, состоящих в отношении следования с протоколом B_k, их предусловия не должны пересекаться.

$$\forall k \quad (\forall i,j \; Suc(B_k,B_i) \;\&\; Suc(B_k,B_j) \;\&\; \sim(P_i \;\&\; P_j))$$

- **Полнота.** Полная система не должна иметь тупиков. Для каждого базового протокола B_i дизъюнкция предусловий P_j протоколов B_j состоящих с ним в отношении следования должна быть равна 1.

$$\forall i \left(\bigvee_{Success(Bi,Bj)} (P_j) = 1 \right)$$

Недетерминизм в требованиях

- R1. The indicator is to be shown when the train is receiving a SDE signal.
- R2. Upon receiving of SDE signal should change operation mode and send shutdown signal to TMS

Что все же произойдет после получения сигнала SDE?

- R1. Upon transition to SLEEP mode processor starts send request for shutdown every 5 second.
- R2. If transition to SLEEP mode was performed with signal SAVE_CONF then processor starts perform saving of current system configuration.

Что должно произойти раньше если это не альтернативы?

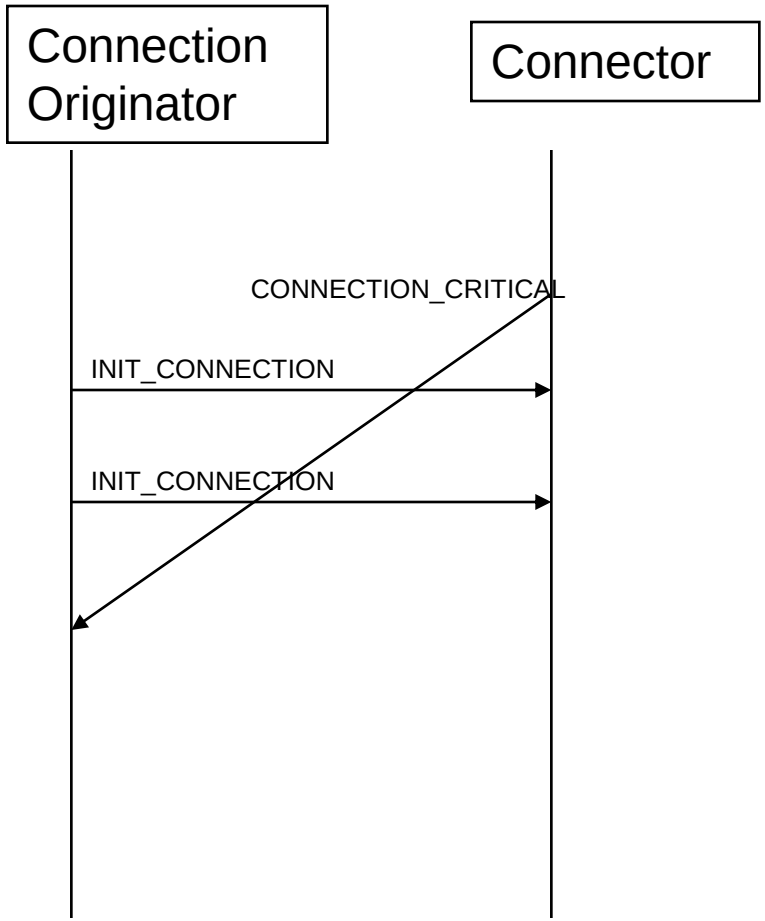
Тупики в требованиях

- R1. After pressing the button, the device sends the signal START_TALK and changes the state to waiting for the response, named WAITING.
- R2. If START_TALK is received, the called device, which was in IDLE state, sends the acknowledgement signal TALK_ACKNOWLEDGEMENT and transfers to the state of receiving call named HEARING.
- R3. After receiving the confirmation signal TALK_ACKNOWLEDGEMENT, the calling device transfers to the state of voice transmission.
- R4. If the button is released, the calling device sends END_TALK signal and changes the state to IDLE.
- R5. After receiving the signal END_TALK the called device, which was in state HEARING, would transfer to the state IDLE.

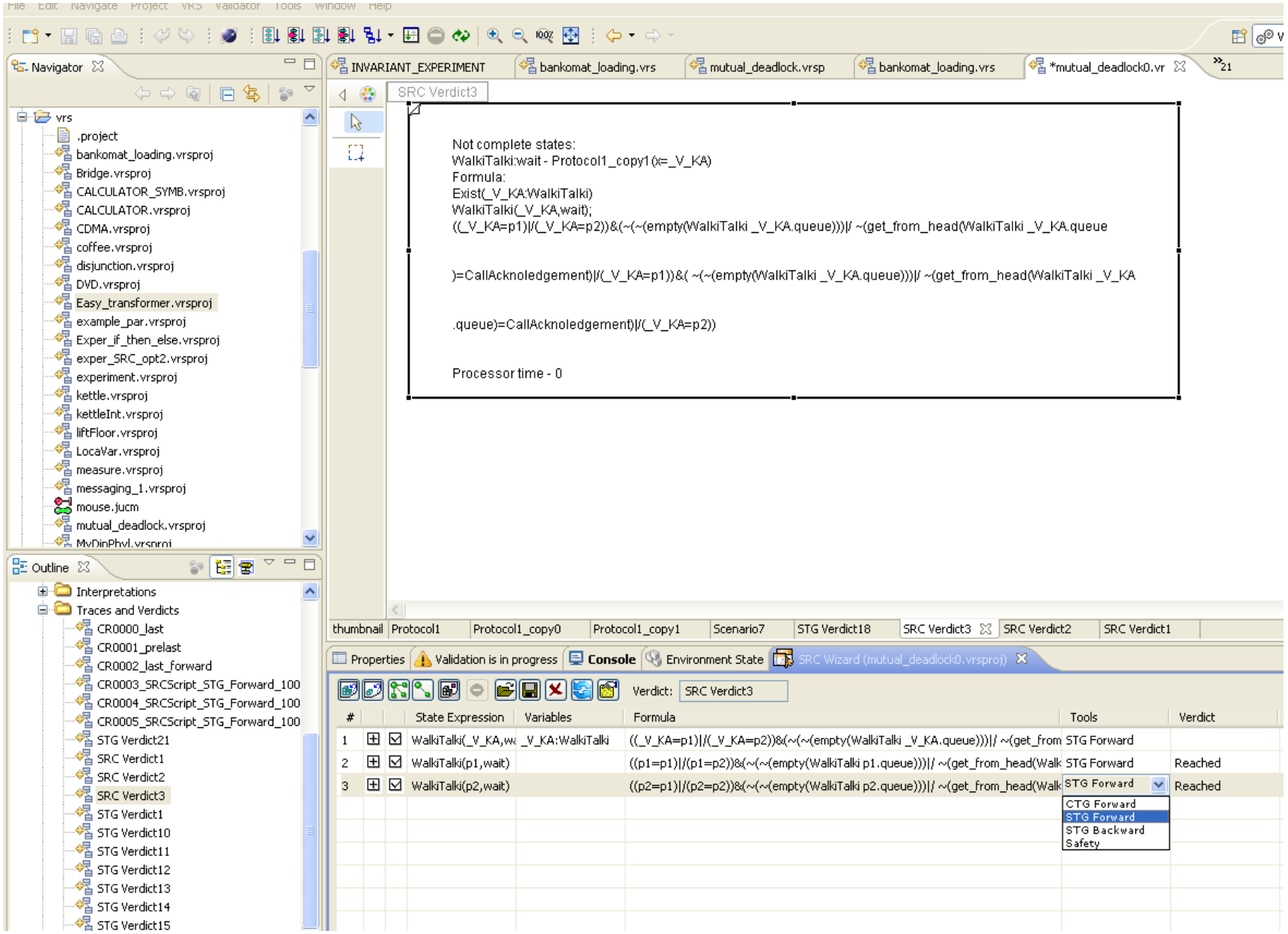
Что произойдет если оба пользователя нажмут кнопку START_TALK?

Свойство безопасности

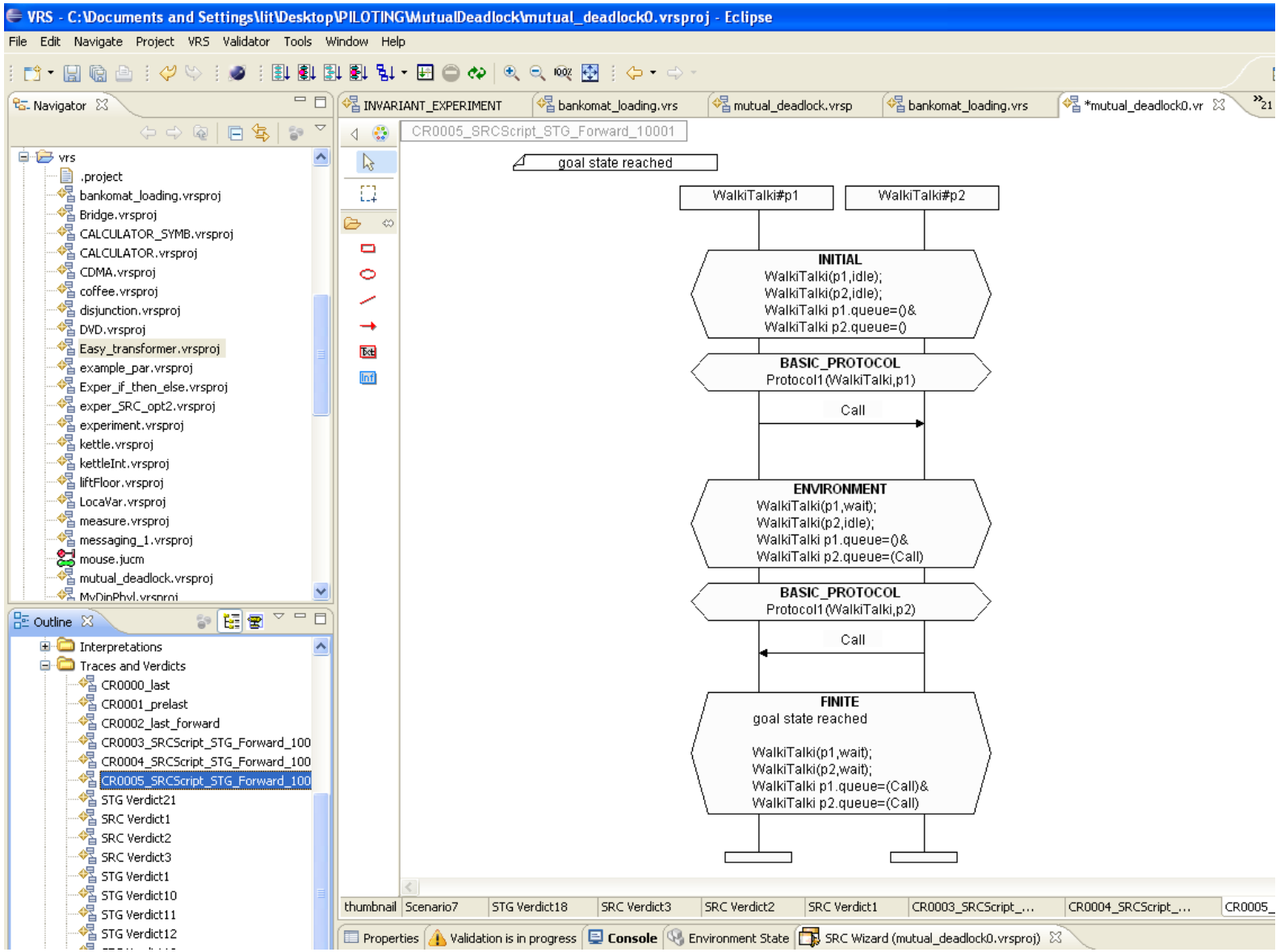
- R1. Originator of connection send to connector signal INIT_CONNECTION which is the command for connection setting.
- R2. Upon obtaining INIT_CONNECTION signal the connector set the new connection.
- R3. If the the critical number of connections was reached then connector send the signal CONNECTION_CRITICAL
- R4. Originator of connection stops send signal INIT_CONNECTION upon receiving of CONNECTION_CRITICAL
- R5. (Safety) The number of connections should not increase the critical values



Статическая проверка требований (вердикт)

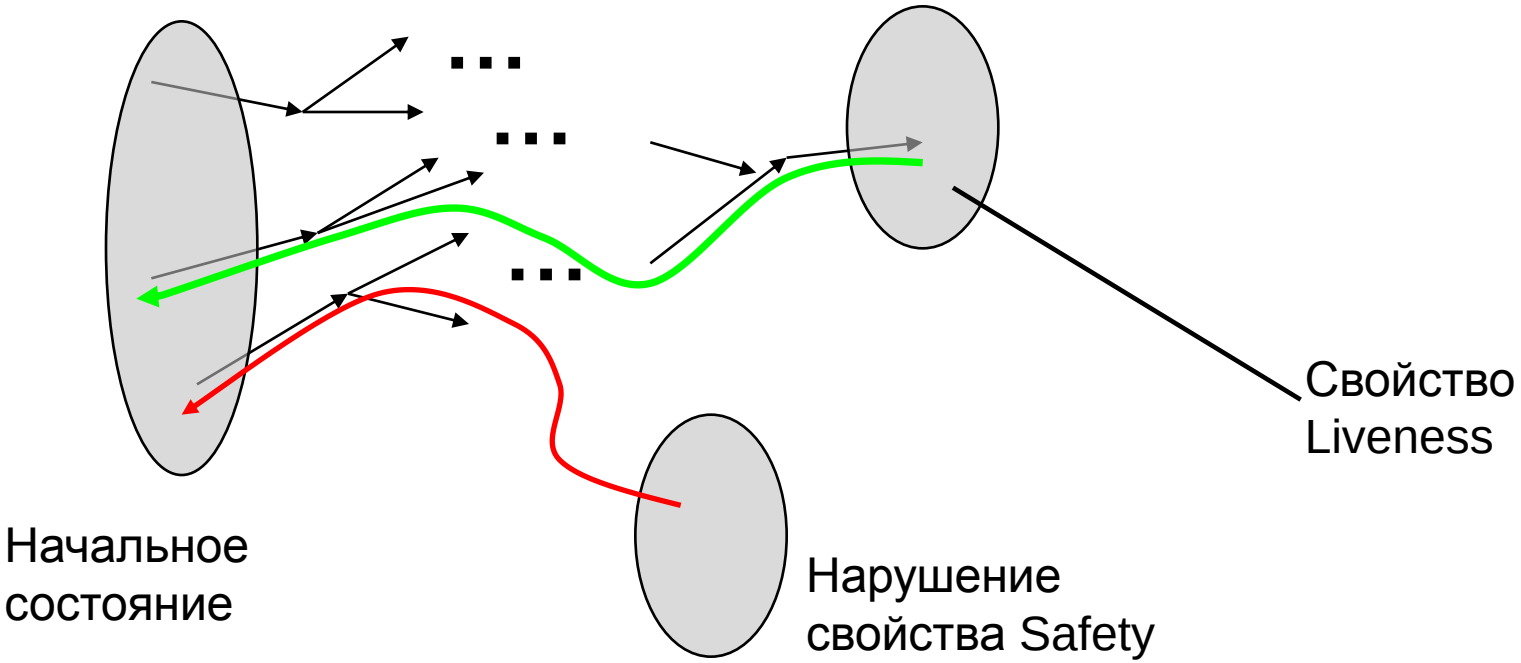


Трасса иллюстрирующая достижимость тупика



Обратное символьное моделирование

- Обратный предикатный трансформер



Верификация телекоммуникационного протокола

Статистика

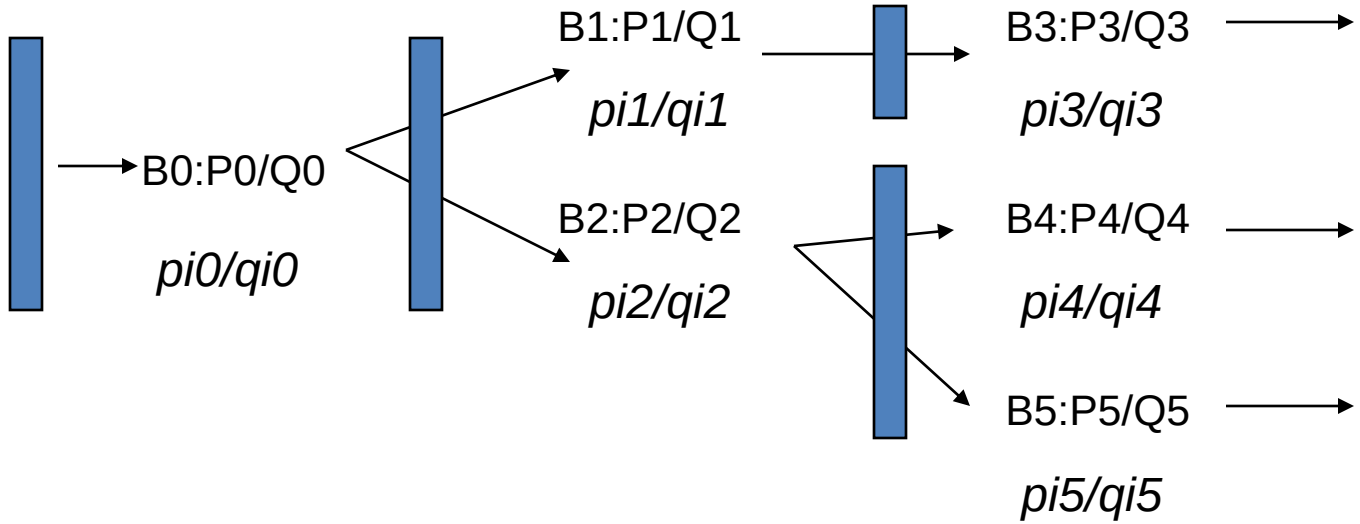
Attribute	Value	Comment
Pages	971	Total number of pages in the source documentation
Requirements	6000	Total number of all requirements in the source documentation
Basic Protocols	558	Total number of basic protocols developed from the behavioral requirements
Trace Space	$7*10^9$	Total number of traces originated from the developed basic protocols
Findings	87	0:Low; 42:Medium; 45:High
Document errors	116	12:Low; 104:Medium; 0:High

Effort in Staff-weeks Spent for	
Developing basic protocols	4.35
Verification	1.0
Total:	**5.35**

Инварианты

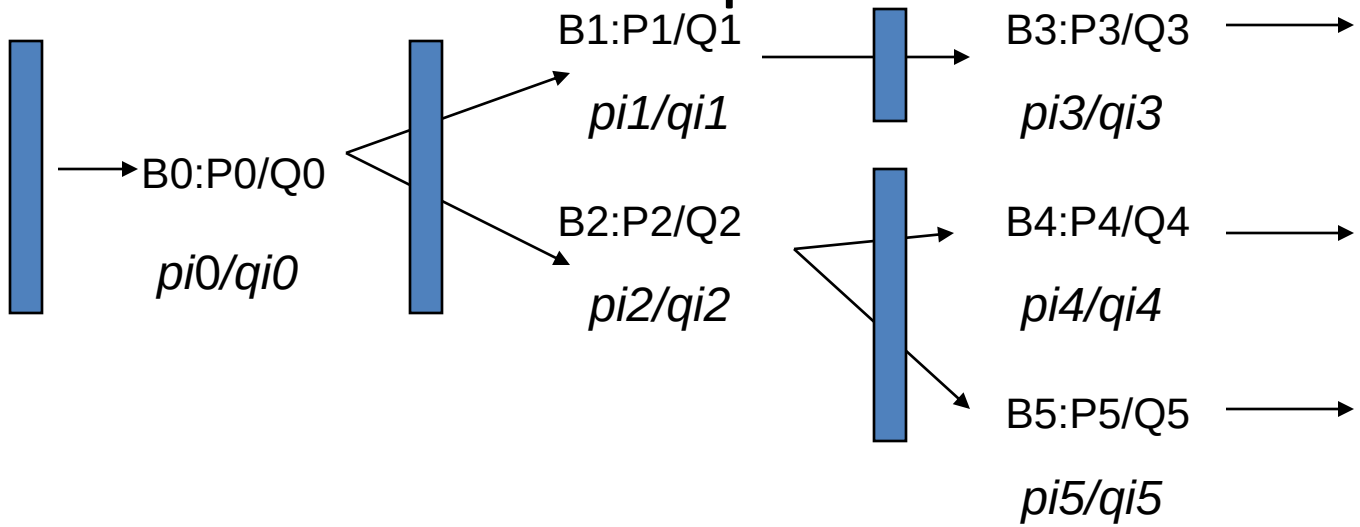
- Свойство среды является преинвариантом (постинвариантом) базового протокола если оно всегда истинно до (после) успешного применения этого базового протокола.
- Сильнейший инвариант является такой инвариант, что все остальные инварианты являются его следствием.

Моделирование и верификация с помощью инвариантов



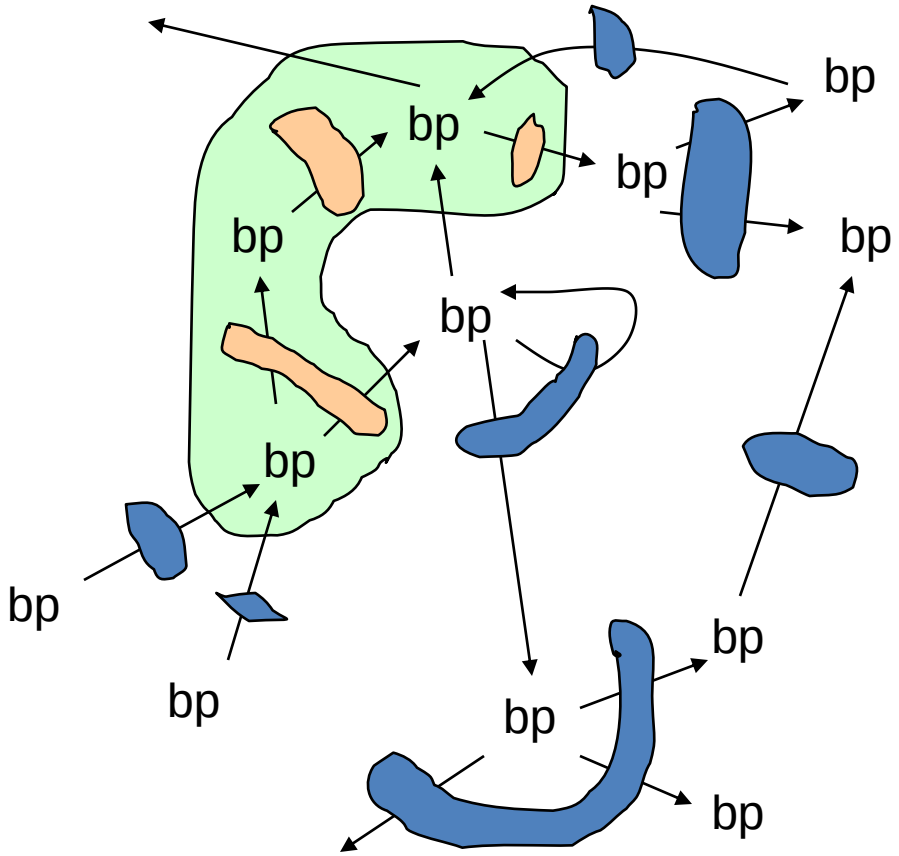
- Если в системе базовых протоколов вычислены пре- и постинварианты для каждого базового протокола, то моделирование сводится к нахождению путей в графе следования базовых протоколов.
- Если искомое свойство является W то достижимость этого свойства может быть определена статически проверкой выполнимости формулы W & qi_n
- Bi = (Pi,Qi) – базовый протокол с предусловием Pi, и постусловием Qi, где построен сильнейший преинвариант pi_n и постинвариант qi_n

Моделирование и верификация с помощью инвариантов



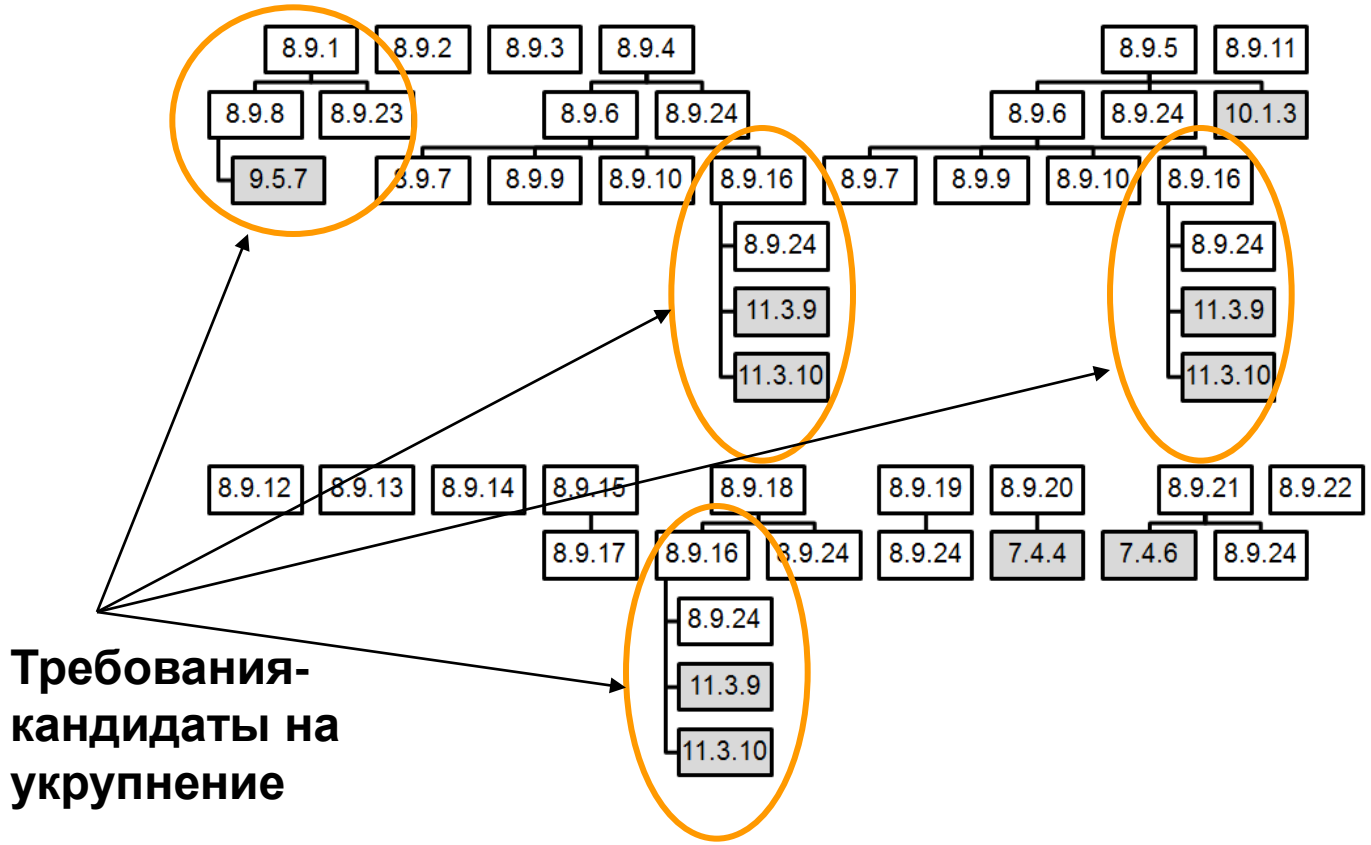
- Если все сильнейшие инварианты вычислить невозможно, то можно определить статически недостижимость свойства W в каждом состоянии ожидания после выполнения базового протокола доказательством невыполнимости формулы W & qi_n

Использование техники укрупнений при верификации

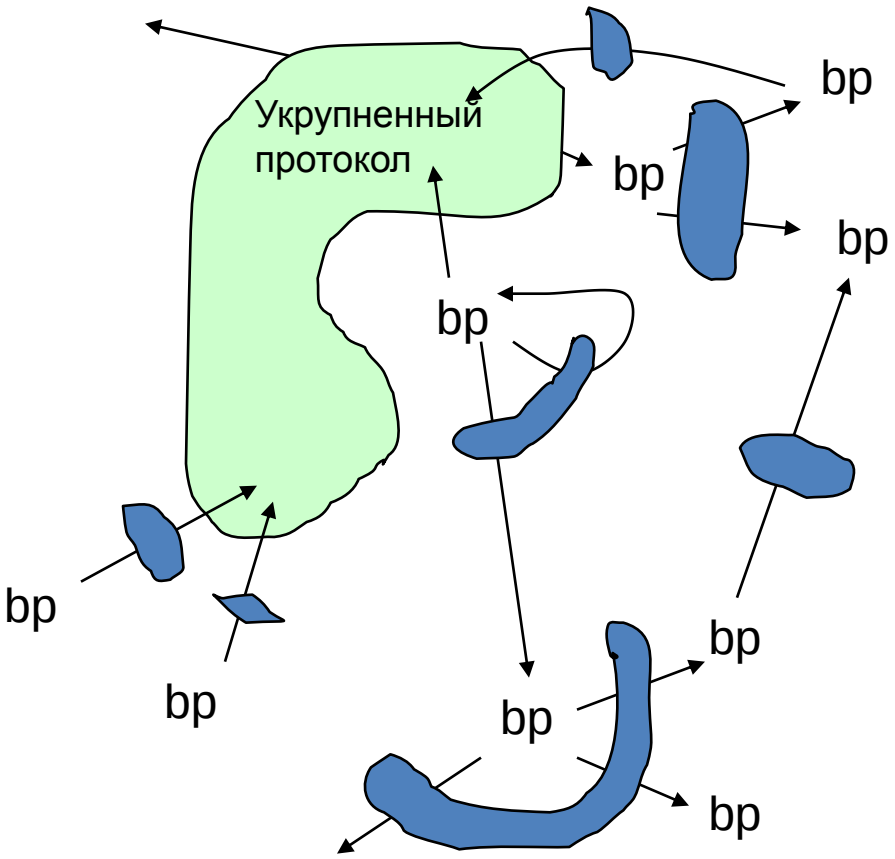


Базовые протоколы могут быть объединены в группы или в один укрупненный протокол, который потом можно использовать в крупношаговом моделировании.

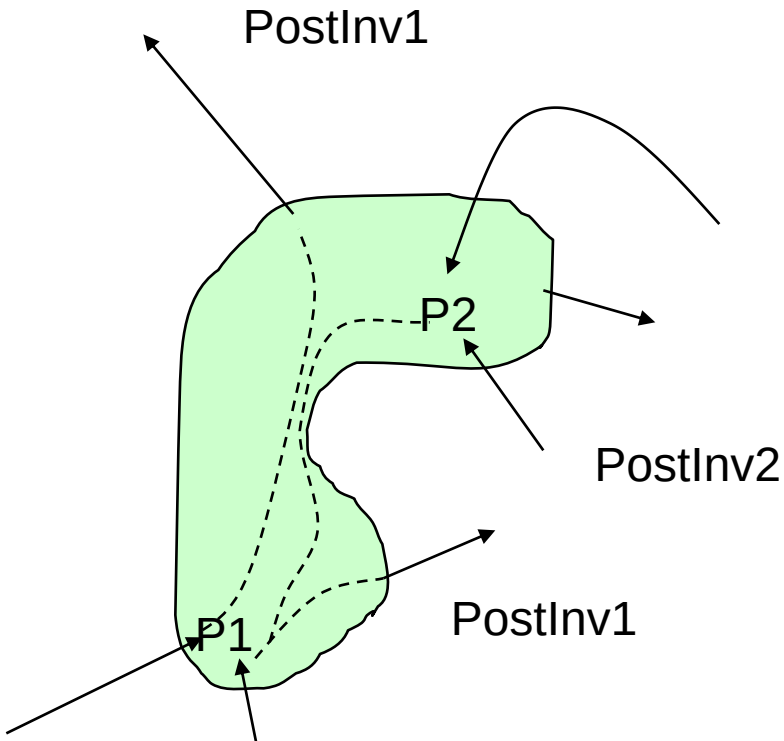
Использование укрупнения в инкрементальной верификации



Критерий укрупнения



Базовые протоколы могут объединяться в группу, если для выходных стрелок из группы на графе следования посчитан постинвариант соответствующего базового протокола



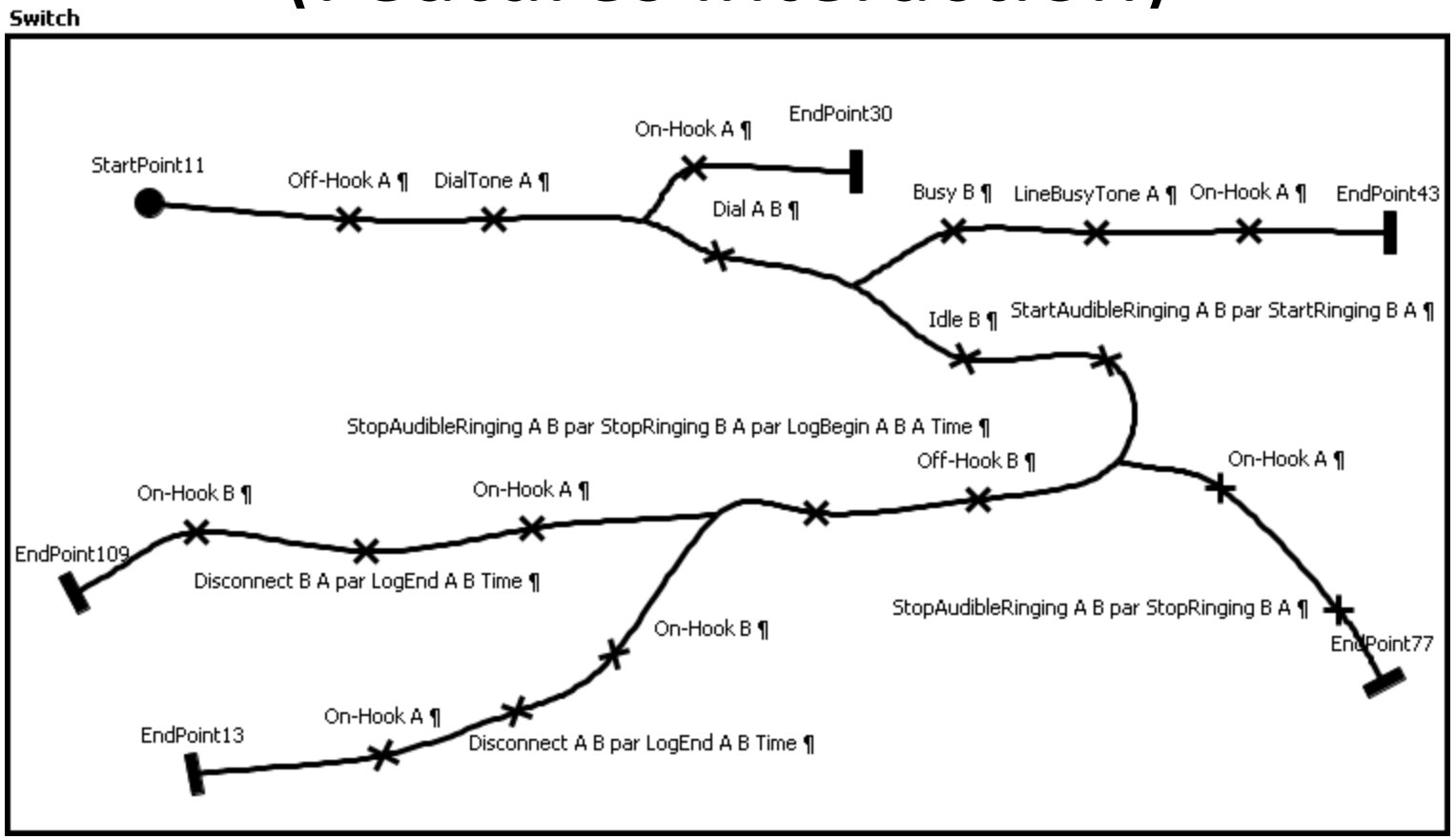
Предусловие: P1 $\vee$ P2

Постусловие: X:bool,Y:bool;

X :=P1;Y := P2

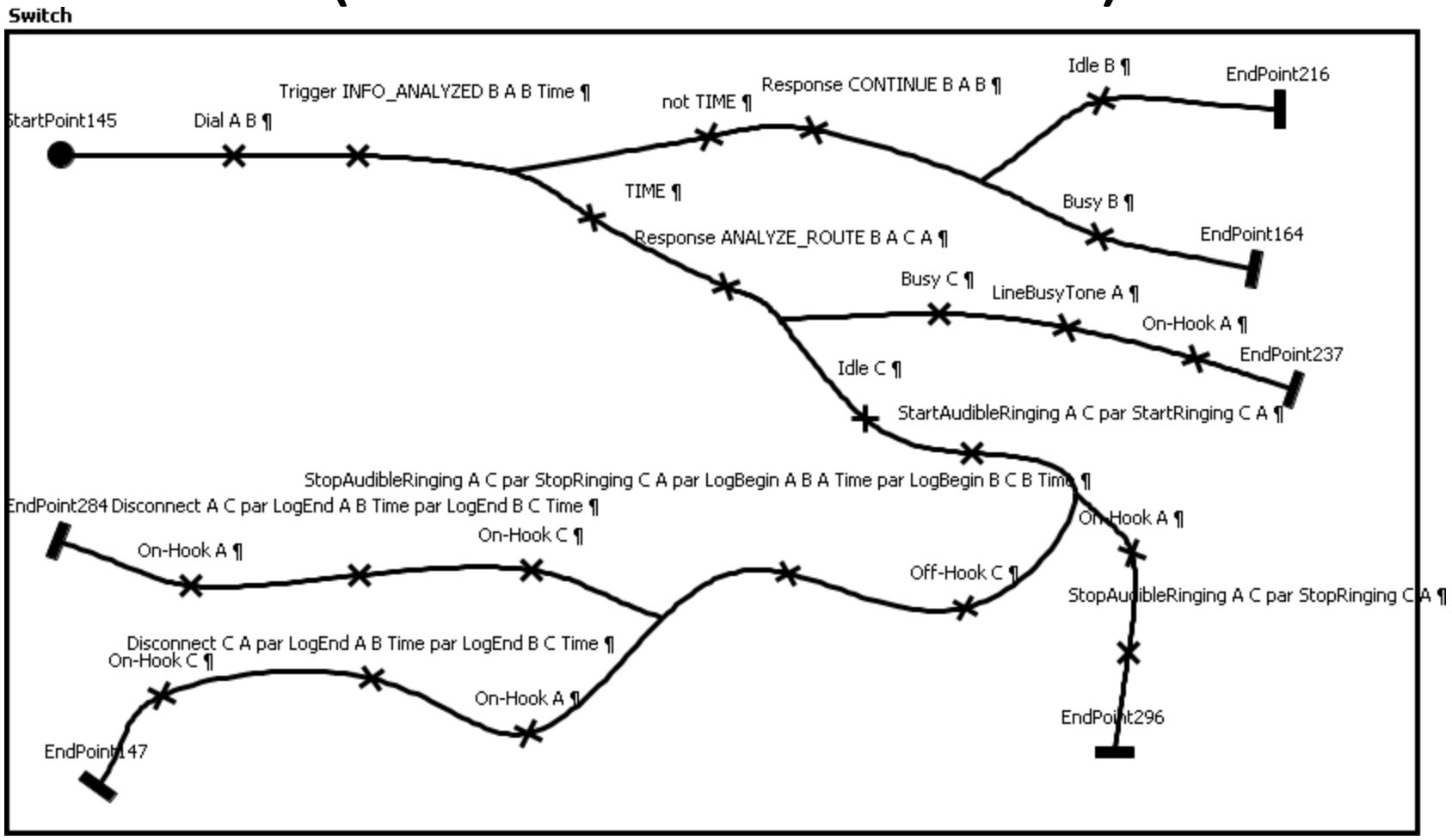
X&(Postinv1 $\vee$ PostInv2) $\vee$ Y&(Postinv2)

Пересечение функциональностей (Features interaction)



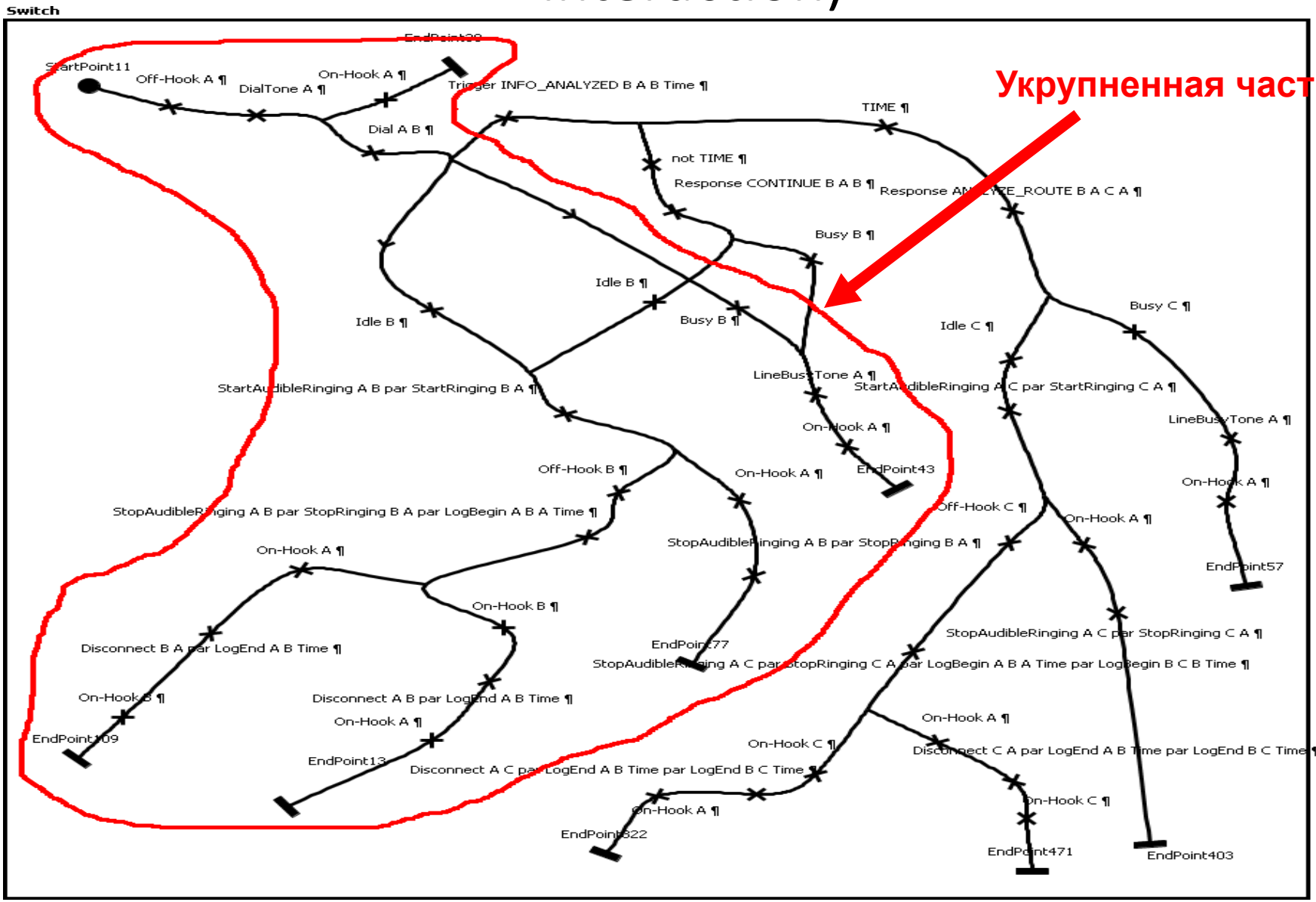
Basic Call feature

Пересечение функциональностей (Features interaction)



IN Freephone Routing

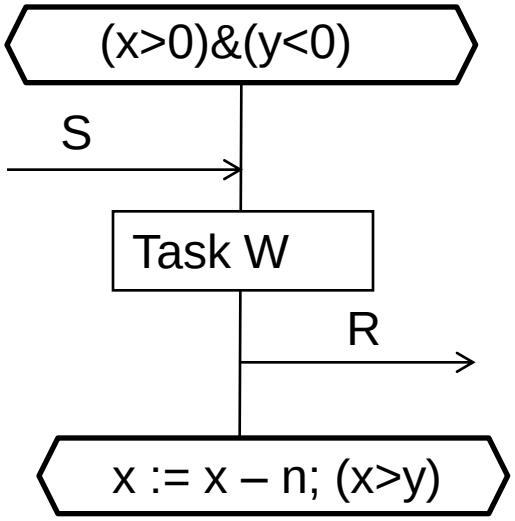
Пересечение функциональностей (Features interaction)



Часть 2
Верификация и тестирование дизайн-моделей и программного кода

Генерация дизайн-модели

- Базовые протоколы могут быть конвертированы в SDL/UML спецификации
  - Агент можно рассматривать как процесс в SDL/UML
  - Состояния ожидания рассматривать как состояния процесса SDL/UML;
  - Триггеры в базовых протоколах рассматривать как сигналы;
  - Изменение среды рассматривать как переход




```
STATE G1;
INPUT S;
DECISION( x>0 & y <0);
        (true):TASK W;
         OUTPUT R;
ENDDECISION;
        TASK(x := x – n);
        ANNO(x>y);
NEXTSTATE G2;
```

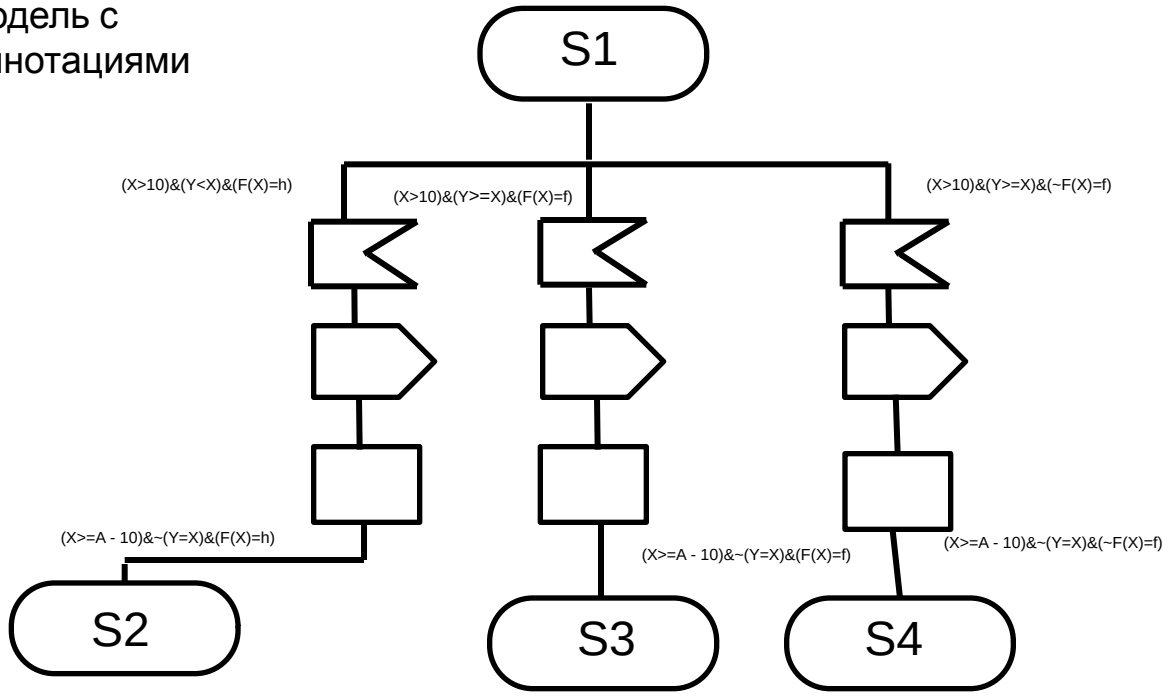
Аннотации в дизайн-модели

Множество инвариантов
Базовые протоколы
UCM

генерация →

SDL/UML модель с аннотациями

```
STATE S1;
INPUT m1;
ANNO ((X>10)&(Y<X)&(F(X)=h);
OUTPUT r1;
TASK L1;
ANNO ((X>= A -10)&~(Y=X)&(F(X)=h);
NEXT S2;
INPUT m2;
ANNO ((X>10)&(Y>=X)&(F(X)=f);
OUTPUT r2;
TASK L2;
ANNO ((X>= A -10)&~(Y=X)&(F(X)=f);
NEXT S3;
INPUT m3;
ANNO ((X>10)&(Y>=X)&~(F(X)=f);
OUTPUT r3;
TASK L2;
ANNO ((X>= A -10)&~(Y=X)&~(F(X)=f);
NEXT S4;
```



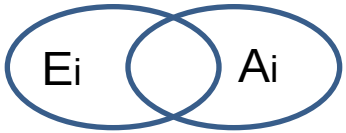
Проверка аннотаций в уточненной дизайн-модели

- Символьное моделирование SDL/UML спецификаций выполняется также с помощью предикатных трансформеров. Моделирование дизайн-спецификаций порождает последовательность символьных состояний среды на каждом переходе SDL/UML : $E_1 \to E_2 \to \ldots \to E_i$

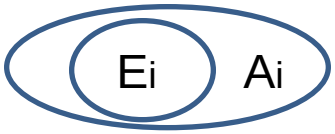
Каждый i-й переход содержит аннотации Ai, где A_i –постинвариант

- Проверка соответствия уточненной модели требованиям

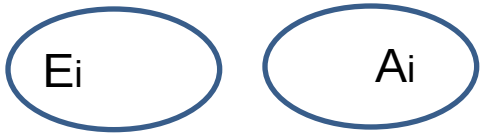
Если $A_i \Rightarrow E_i$, то поведение модели соответствует начальным требованиям иначе модель содержит сценарий не описанный в спецификациях. Соответствующая трасса ведущая к недопустимому множеству значений атрибутов должна быть построена.



Неописанное поведение существует, Ai – произвольный постинвариант



Поведение соответствует требованиям, Ai – сильнейший инвариант



Не соответствует требованиям, Ai – произвольный постинвариант

Проверка свойств в SDL/UML

- Проверка свойства безопасности S может быть проверено символьным моделированием

~S & E_i – должно быть невыполнимым

Если ~S & E_i выполнимо, значит мы можем построить трассу ведущую от начального состояния к нарушению с использованием того же обратного моделирования

- Инварианты в SDL

Множество инвариантов может быть использовано при верификации C/C++ программ в качестве аннотаций

Аннотации в сгенерированном C/C++ коде

C/C++ код:

```
input(x);
if (x>0)&&(x<10)
        C = C + x;
If (C > 10)
        C = C – 2;
/*Anno: (C>x) */
output(C);
```

Символьное моделирование C/C++:

(C > 0)

(C > x) & (x>0)&(x<10)

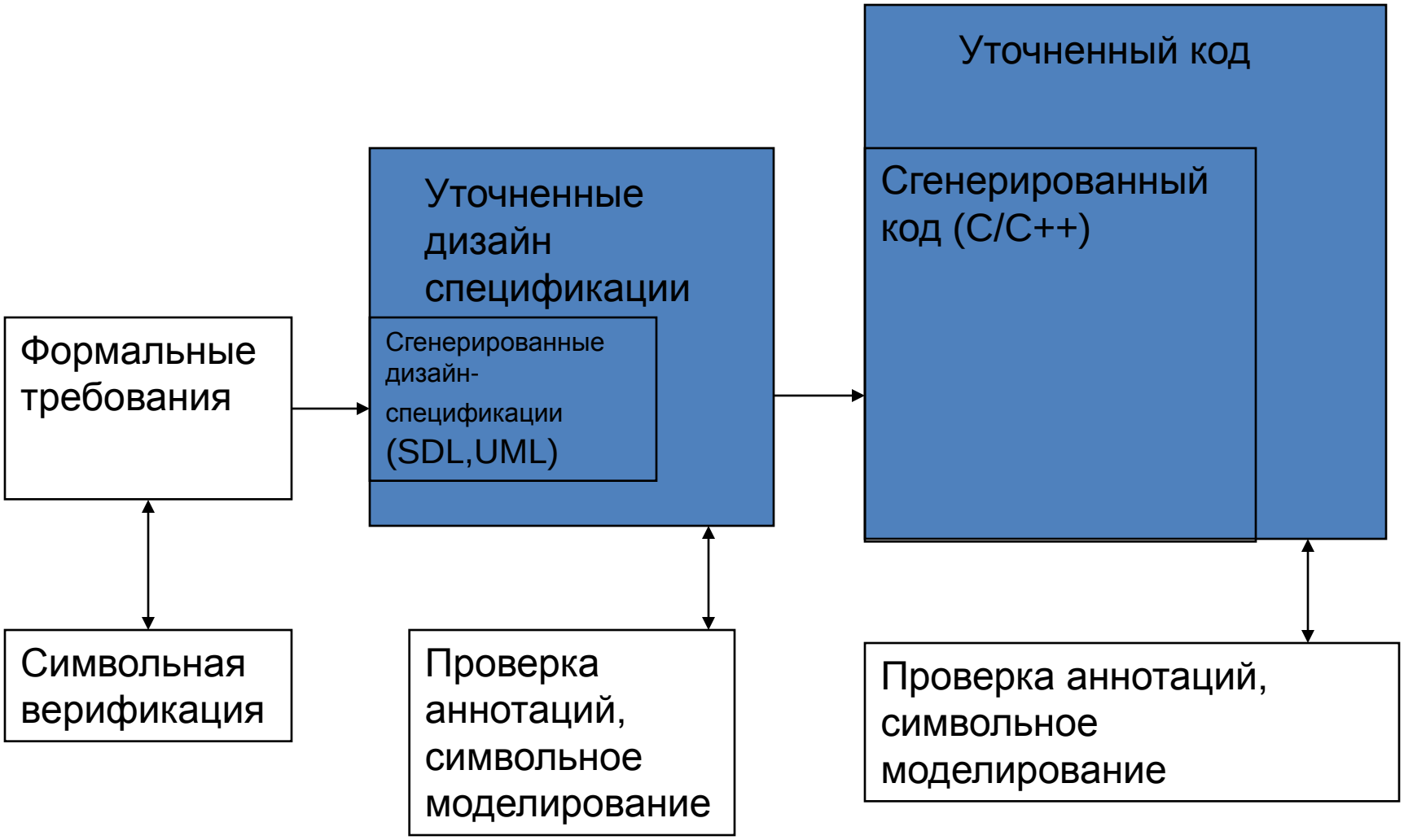
(C > x - 2) & (C > 8) & (x>0)&(x<10)

Проверка совместимости с дизайн-моделью осуществляется символьным моделированием с проверкой выполнимости следующей формулы:

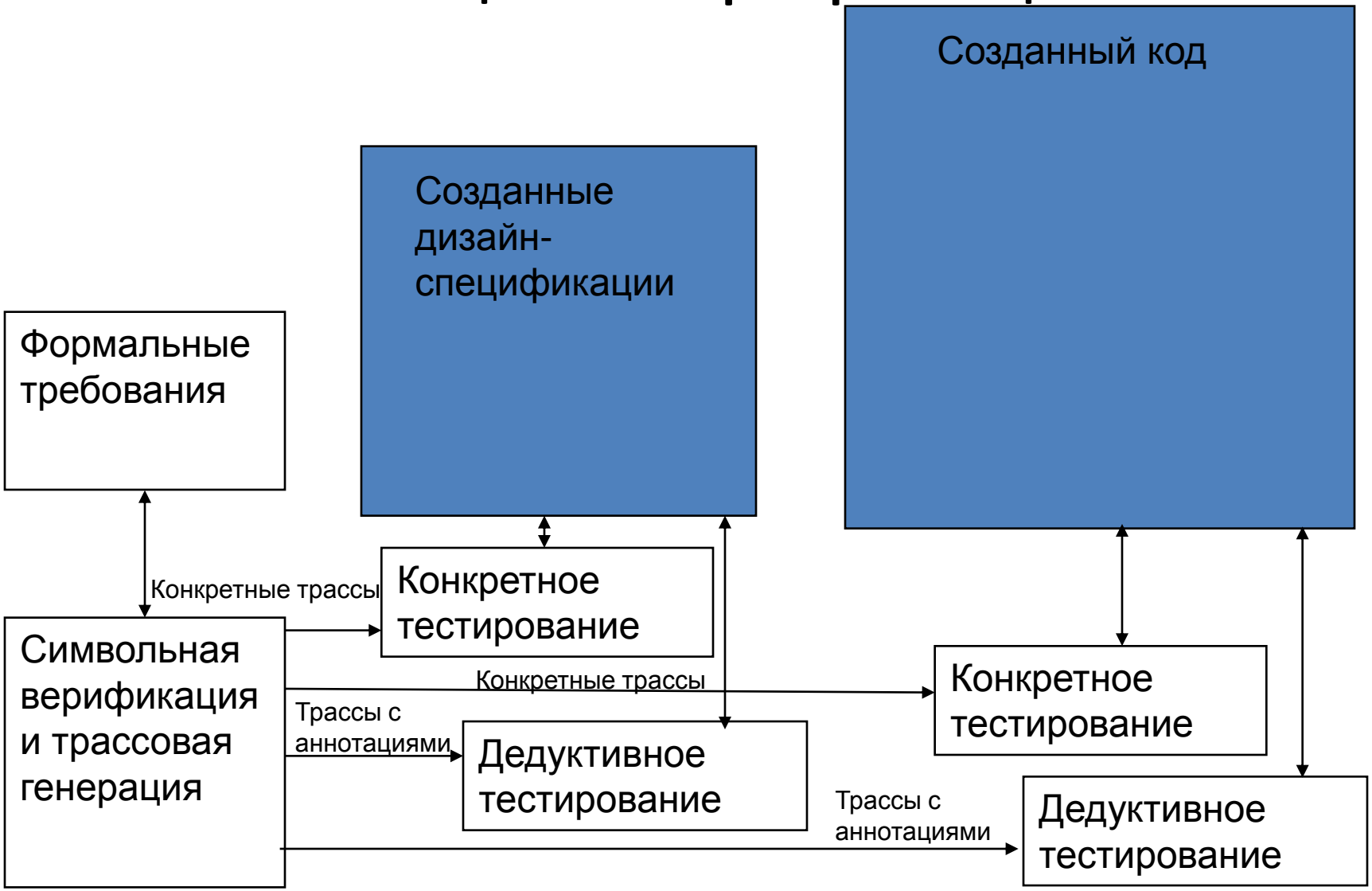
(C>x) => (C > x - 2) & (C > 8) &(x>0)&(x<10)

Доказательство аннотаций может выполнятся методом Флойда

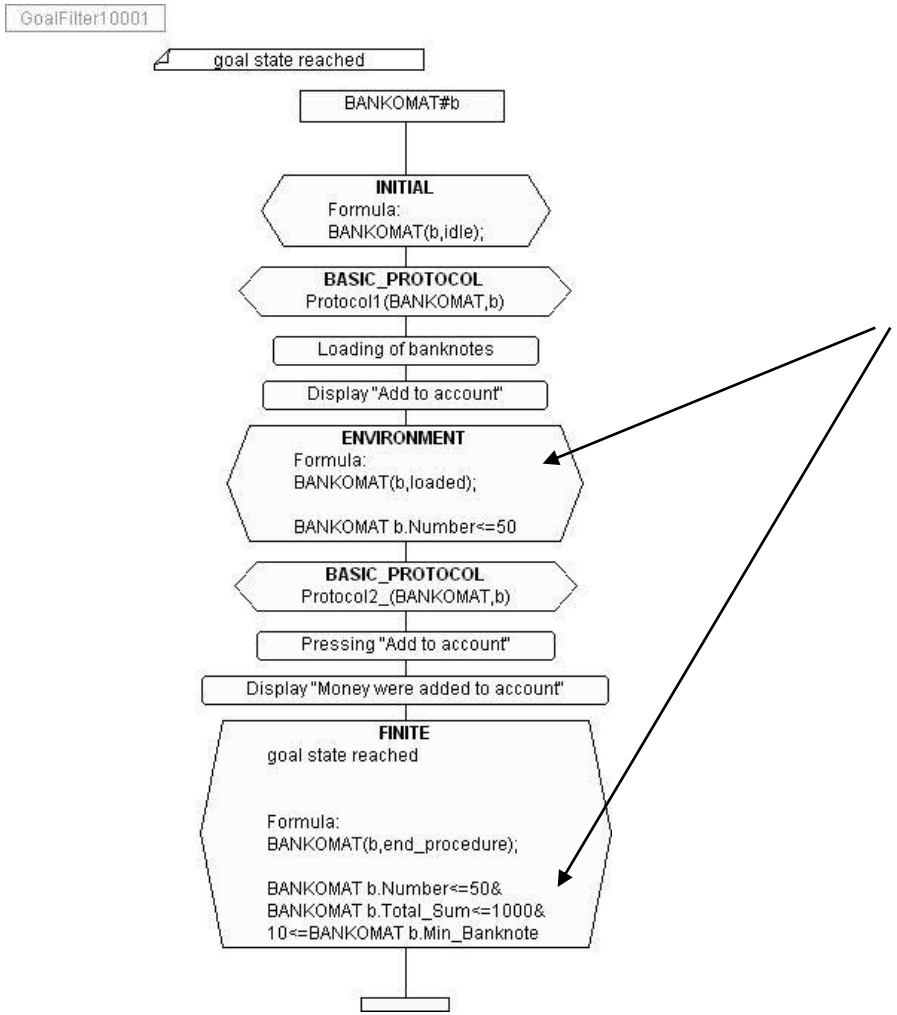
Полный цикл верификации



Полный цикл верификации

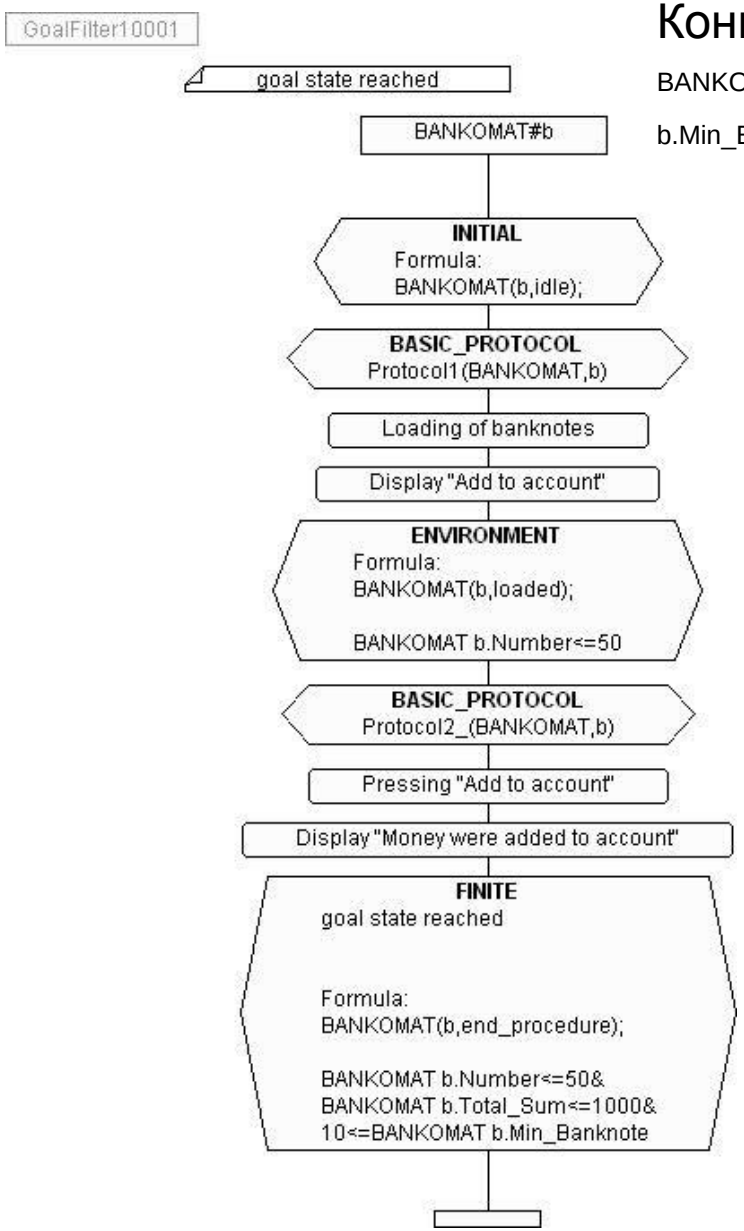


Дедуктивное тестирование



При символьном моделировании спецификаций (базовых протоколов, SDL/UML модели мы получаем символьную трассу в которой покрываются сценарии для множеств атрибутов посредством символьного состояния среды.

Дедуктивное тестирование



Конкретное начальное состояние трассы:

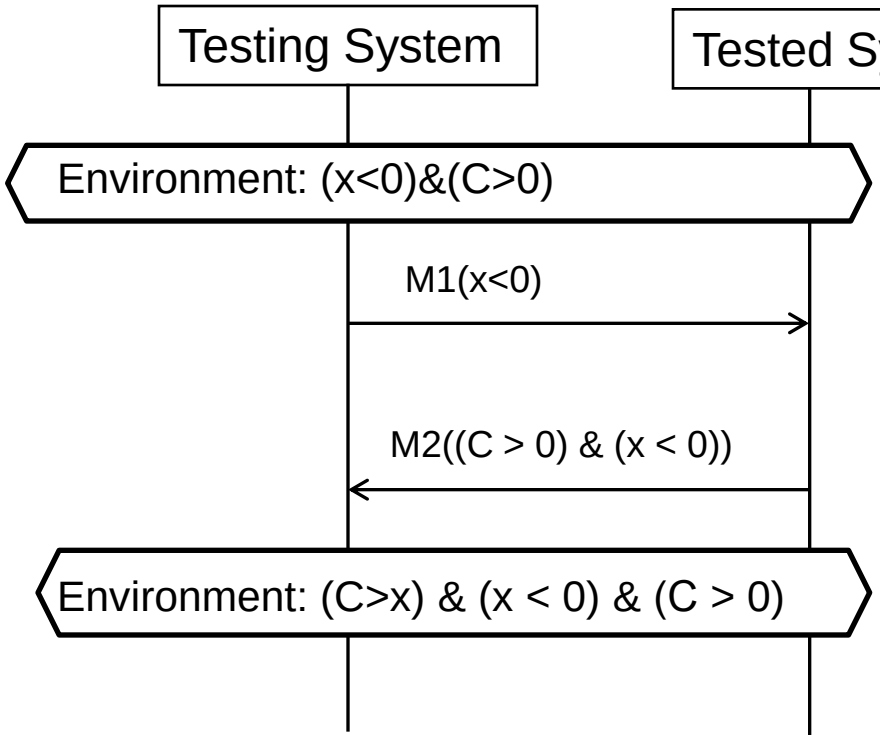
BANKOMAT b.Number = 10 & BANKOMAT b.Total_Sum = 500 & BANKOMAT b.Min_Bank_Note = 100 & My_Money = (100,100)

Выбрав конкретные значения в заключительном состоянии среды, мы можем построить конкретную трассу из символьной

Конкретное заключительное состояние трассы

BANKOMAT b.Number = 10 & BANKOMAT b.Total_Sum = 500 & BANKOMAT b.Min_Bank_Note = 2

Дедуктивное тестирование



C код:	**Символьное моделирование C:**
	(C > 0)
input();	/* ввод формулы x<0 */
if (x>0)	
C = C + x;	
	(C > 0) & (x < 0)
Output(*(C > 0) & (x < 0)*);	

Проверка совместимости с тестом:

(C > x) & (x <0) & (C > 0) <=> (C > 0) & (x < 0)

Дедуктивное тестирование должно использоваться наряду с конкретным тестированием

Спасибо!