Формальные методы в программной инженерии
Спецификация и верификация моделей в системе Event-B

магистерский курс, 2021

Шелехов Владимир Иванович,
зав. лаб. Системного программирования ИСИ, к.т.н.,
Институт Систем Информатики, Новосибирск,
Новосибирский Государственный Университет,
vshel@iis.nsk.su, р.т. (383) 330-27-21, ИСИ СО РАН, к.269

23.11.21 ФМ5 Лекция 1

Язык спецификаций Event-B

http://www.event-b.org/ - сайт **Event-B.org**

Язык Event-B и платформа Rodin

B метод – предшественник Event-B

Язык спецификаций Event-B: контекст, машина, событие, действие, уточнение.

Event-B ↔ Автоматное программирование

Предикаты, числа, множества

Отношения, функции. Теории

Правила корректности

Язык Event-B и платформа Rodin

Назначение: спецификация и верификация моделей в системной и программной инженерии

Применение: автоматизация в метро, ж.-д. транспорте

Язык спецификаций **Event-B**: статическая и динамическая части, язык теории множеств.

Метод уточнений.

Платформа **Rodin**: генерация формул корректности инвариантов, система автоматического и интерактивного доказательства, model-checker, набор инструментов доказательства, SMT-решатели, аниматор

Построена на базе среды **Eclipse**

Более 50 расширений с помощью плагинов

http://www.event-b.org/ - сайт **Event-B.org**

B метод – предшественник Event-B

14-я полностью автоматическая линия парижского метро (без машиниста) была разработана с применением формального метода **B**, на основе которого позже разработан метод **Event-B**.

14-я линия запущена в 1998г.

- Была написана и полностью доказана модель на языке B размером свыше 110 000 строк
- На её основе было сгенерировано 86 000 строк кода на языке **Ada**
- С момента завершения доказательства не было найдено ни одной ошибки – даже после дополнительных работ по верификации и валидации системы

Язык спецификаций Event-B

Event-B применяется для разработки и верификации моделей в системной и программной инженерии только в классе программ-процессов (систем управления).

Язык **Event-B:**

- типы данных – язык теории множеств.
- контексты – статическая часть
- машины – динамическая (автоматная) часть

Метод уточнений.

Контекст

Неизменяемая часть спецификации: определения *констант*, *множеств* и *аксиом.*

CONTEXT < context_identifier >

EXTENDS < context_identifier_list >

SETS

< set_identifier_list >

CONSTANTS

< constant_identifier_list >

AXIOMS

< label >: < predicate >

. . .

TEOREMS

< label >: < predicate >

. . .

END

CONTEXT
ColoursExample
SETS
Colours
CONSTANTS
Red
Green
AXIOMS
axm1: Colours={Red, Green}
axm2: Green≠Red

Машина

Динамическая часть: *переменные*, *инварианты*, *события*.
Константы и аксиомы импортируется из контекстов.

MACHINE < machine_identifier >
REFINES < machine_identifier >
SEES < context_identifier_list >
VARIABLES < variable_identifier_list >
INVARIANTS < label >: < predicate > …
THEOREMS < label >: < predicate > …
VARIANT < variant >
EVENTS < event_list >
END

Событие **INITIALISATION**

Событие состоит из:

- *параметров*
- *охранных условий* (guards)
- *действий* (actions)

Событие

Параметры, охранные условия, действия.

EVENT < event_identifier >
STATUS {ordinary, convergent, anticipated}
REFINES < event_identifier_list >
ANY < parameter_identifier_list >
WHERE < label >: < predicate > …
WITH < label >: < witness >
THEN < label >: < action > …
END

Действие

< label >: < action >

Детерминированные, недетерминированные действия

Виды действий:

< variable_identifier > := < expression >

< identifier > (< expression_1 >) := < expression_2 >

< variable_identifier_list > :| < before_after_predicate >

< variable_identifier > :∈ < set_expression >

Примеры:

act1 : x, y :| x′ > y ∧ y′ > x′ + z

act1 : x :∈ A ∪ {y}

эквивалентно

act1 : x :| x′ ∈ A ∪ {y}

Уточнение

Уточнение (refinement) . Большая и сложная спецификация реализуется в виде последовательности спецификаций, где каждая следующая уточняет предыдущую.

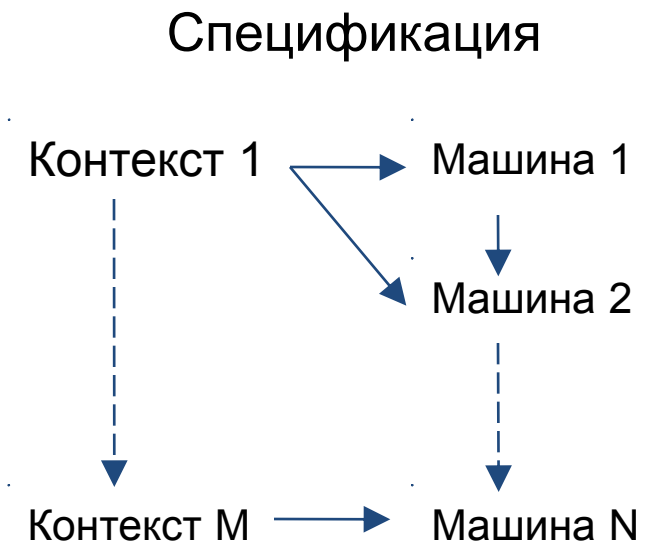
Конкретная машина B уточняет абстрактную машину A ≅ поведение машины B согласовано с поведением машины A.

Склеивающий (gluing) инвариант содержит переменные конкретной и абстрактной машины – устанавливает соответствие между B и A.

Событие абстрактной машины может быть уточнено одним или несколькими событиями конкретной машины. Конкретное событие случается лишь когда случается абстрактное событие (усиление гарда). Другие конкретные события не должны менять состояния абстрактной машины.

1. Уточнение детализацией данных
2. Уточнение введением дополнительных данных.

Уточнение


Спецификация
Контекст 1
Машина 1
Машина 2
Контекст M
Машина N

Event-B ↔ Автоматное программирование

Автоматная программа – набор *правил* вида:

M: <условие$_1$>,…, <условие$_n$> → <действие$_1$>, …, <действие$_m$> #L

Машина на языке Event-B – недетерминированная композиция:

process **Z**(…); {**Cy: [nA:] <правило$_1$> | … | [nK:] <правило$_k$> #Cy**}

Каждое правило (сегмент) реализуется событием Event-B.

Отличия:

- **Управляющие состояния устраняются по Switch-технологии**
- **Нет сообщений**
- **Нет времени**
- **Нет вызовов процессов в действиях**
- **Цепочка действий в правиле – это мультиприсваивание. Суперпозиция недопустима (в т.ч. локалы); взамен можно использовать действие вида** < ident_list > :| <before_after_predicate>

Нет параллельной композиции, нет иерархии процессов. Нет определений предикатов. Уточнение – единственный вид композиции вместо объектно- и аспектно-ориентированной композиции.

Язык Event-B. Предикаты

	ASCII	Логика первого порядка
$\bot$	false	
$\top$	true	
$P \land Q$	P & Q	
$P \lor Q$	P or Q	
$P \Rightarrow Q$	P => Q	
$P \Leftrightarrow Q$	P <=> Q	
$\neg P$	not P	
$\forall z \cdot P \Rightarrow Q$	!z. P => Q	Тип z выводим из P
$\exists z \cdot P \land Q$	#z. P & Q	Тип z выводим из P
$E = F$	E = F	
$E \neq F$	E /= F	

Язык Event-B. Числа

	ASCII	
Z	INT	
N	NAT	
$\mathbf{N_1}$	NAT1	$N_1 = N\setminus\{0\}$

$m \times n$	m * n
$m \geq n$	m >= n
$m \leq n$	m <= n

$m + n$ $m - n$ m / n $m > n$ $m < n$

$m \bmod n$ $m .. n$ $= \{ i \mid m \leq i \wedge i \leq n \}$

$\min(S)$ $\max(S)$ где $S \subset Z$

Язык Event-B. Множества

	ASCII
{E}	{E}
{E,F}	{E, F}
∅	{}
{z·P | F}	{z . P | F} значения терма F, для которых P истинна
{ x | P }	{ x | P } = {x·P | x}
S ∪ T	S \/ T
S ∩ T	S /\ T
S \ T	S \ T
E ↦ F	E |-> F упорядоченная пара (E, F) – это список
S × T	S ** T = {x ↦ y | x ∈ S ∧ y ∈ T}
P(S)	POW(S) = {s | s ⊆ S}
card(S)	card(S) число элементов
finite(S)	finite(S)
partition(S, x, y)	partition(S, {A}, {B}, {C}) ~ **type** S = **enum** A, B, C

Язык Event-B. Множества

	ASCII
$E \in S$	E : S
$E \notin S$	E /: S
$S \subseteq T$	S <: T
$S \not\subseteq T$	S /<: T
$S \subset T$	S <<: T
$S \not\subset T$	S /<<: T

Язык Event-B. Логический тип

BOOL – тип множества {FALSE,TRUE}

Язык Event-B. Отношения. Функции

ASCII

$E \mapsto F$ E |-> F упорядоченная пара

$S \times T$ S ** T = $\{x \mapsto y \mid x \in S \land y \in T\}$

$S \leftrightarrow T$ S <-> T = **P**(S × T) множество отношений

$\forall r \cdot r \in S \leftrightarrow T \Rightarrow dom(r) = \{x \cdot (\exists y \cdot x \mapsto y \in r)\}$ область определения (domain)

$\forall r \cdot r \in S \leftrightarrow T \Rightarrow ran(r) = \{y \cdot (\exists x \cdot x \mapsto y \in r)\}$ область значений (range)

$S \nrightarrow T$ S +-> T частичная функция: любому элементу домена соответствует только один элемент из области значений

$S \rightarrow T = \{f \cdot f \in S \nrightarrow T \land \mathrm{dom}(f) = S\}$ S --> T тотальная функция

f(E) вызов функции

Краткое описание языка Event-B:

https://wiki.event-b.org/images/EventB-Summary.pdf

Язык Event-B. Теории

Теория составляется из следующих частей:

- импортируемые теории
- типы – параметры теории множества
- алгебраические типы (data types) : конструкторы, поля
- операции: префиксные, инфиксные
- аксиоматическое определение
- теоремы
- правила доказательства (proof rules) rewrite rules

Библиотека: reals, lists, queues, seqs, binary trees, orders, ….

Теории в Event-B. Руководство пользователя:

https://wiki.event-b.org/images/Theory_Plugin.pdf

Правила корректности спецификации

event evt any x where G(s, c, v, x) then v :| BA(s, c, v, x, v′) end

s – мн-ва, c – константы, v – переменные, BA – before-after предикат

A(s, c) – аксиомы и теоремы, I(s, c, v) – инварианты.

1. Событие evt сохраняет инвариант inv(s, c, v) – evt /inv/INV :

A(s, c) & I(s, c, v) & G(s, c, v, x) & BA(s, c, v, x, v′) ⊢ inv(s, c, v′)

event evt any x where G(s, c, v, x) then act: v :| BA(s, c, v, x, v′) … end

2. Выполнимость действия act в событии evt – evt /act/FIS :

A(s, c) & I(s, c, v) & G(s, c, v, x) ⊢ ∃v′·BA(s, c, v, x, v′)

event evt0 any x where grd : g(s, c, v, x) . . . then . . . end

event evt refines evt0 any y where H(y, s, c,w) with x : W(x, s, c,w, y) then … end

3. Усиление гарда grd в событии evt – evt /grd/GRD :

A(s, c) & I(s, c, v) & J(s, c, v, w) & H(y, s, c,w) & W(x, s, c,w, y) ⊢ g(s, c, v, x)

I(s, c, v) – инварианты абстр. evt0 , J(s, c, v, w) – инв-ты конкретного evt

Правила корректности

event evt0 any x where . . . then act: v :| BA1(s, c, v, x, v′) end

event evt refines evt0 any y where H(y, s, c,w)

with x : W1(x, s, c,w, y,w′) v′ : W2(v′, s, c, w, y, w′) then w :| BA2(s, c, w, y, w′) end

s – мн-ва, c – константы, v – переменные, BA – before-after предикат,

A(s, c) – аксиомы и теоремы, I(s, c, v) – абстрактные инварианты,

J(s, c, v, w) – конкретные инварианты, v и w – различны.

4. Согласованность события evt с абстр. действием act в – evt/act/SIM :

A(s, c) & I(s, c, v) & J(s, c, v, w) & H(y, s, c, w) & W1(x, s, c, w, y, w′) &

W2(v′, s, c, w, y, w′) & BA2(s, c, w, y, w′) ⊢ BA1(s, c, v, x, v′)

5. WF – well-definedness – правильность конструкции