ЛЕКЦИЯ 7
ОСНОВИ НА ЕЗИКА ЗА ОПИСАНИЕ НА МОДЕЛИ SIMPLEX MDL

- **Структура на основните компоненти**
- **Алгебрични уравнения**
- **Събития**
- **Повишаване на времето в дискретно-времеви модели**
- **Разделяне на пространството на състоянията**
- **Масиви**

ВЪВЕДЕНИЕ

- **Запознаване с най-важните елементи на моделния описателен език SIMPLEX MDL:**
- **Структура на основните компоненти;**
- **Алгебрични уравнения;**
- **Събития;**
- **Разделяне на пространството на състоянията;**
- **Масиви.**

Структура на основните компоненти

Основните компоненти са основните, изграждащи блокове в Simplex3. Те съдържат описанието на динамичното поведение на модела. Компонентите от по-високо равнище свързват основните компоненти, за да се оформи структура, но те не съдържат описание на динамично поведение.

Структура на основните компоненти

Моделът CedarBog.

Моделът CedarBog описва затлачването на едно езеро от време на време. Моделът е описан първо в R. B. Williams, в статията му „Компютърна симулация на енергиен поток в езерото Кедър“, списание „Системен анализ и симулация в Екологията“, Academic Press 1971 г.

Първоначално моделът използва три променливи на състояния:

p: растения (plants);

h: тревопасни животни (herbivores);

c: месоядни животни (carnivores).

Структура на основните компоненти

- Променливите имат мерни единици „Тонове биомаса“.

Мъртвата органична материя, която е разположена на дъното езерото се представя с o.

Загубата на биомаса в околната среда е представена от променливата e.

Моделът се описва от пет променливи: p, h, c, o и e.

Структура на основните компоненти

Енергията от слънчева светлина се означава със sun.

Тя се описва от следното алгебрично уравнение:

$Sun = 95.9 * (1 + (0.635 * \sin(2 * \pi * T)))$.

Мерните единици за слънчева радиация са kJ/m^2.

Структура на основните компоненти

Слънчевата радиация води до увеличаване на биомасата на растенията за единица време. Превръщащият фактор Bio_Fac е необходим за да превръща слънчевата светлина в увеличаваща се растителна биомаса за единица време:

sun_bio = sun * Bio_Fac

Структура на основните компоненти

Превръщащия фактор Bio_Fac се дава от уравнението:

$$Bio\_Fac = \frac{sun\_bio}{sun} = \frac{\frac{t}{a}}{\frac{kJ}{m^2}} = \frac{\frac{10^3 * kg}{3{,}15 * 10^7 * s}}{\frac{10^3 * Nm}{m^2}} = \frac{\frac{kg}{3{,}15 * 10^7 * s}}{\frac{kg * m * m}{s^2 * m^2}} = \frac{s}{3{,}15 * 10^7} = 3{,}17 * 10^{-8} s$$

$$= 3{,}17 * 10^{-8} s = 3{,}17 * 10^{-8} * (3{,}17 * 10^{-8}) * a \approx 10^{-15} * a$$

Това превръщане лесно се осъществява автоматична от Simplex3, ако единиците за Bio_Fac са дадени като (t * m^2) / (a * kJ). В описанието на модела това е записано като:

Bio_Fac (REAL [t * m^2) / (a * kJ)]): = 1[(t * m^2) / [(a * kJ)]

Структура на основните компоненти

Следните уравнения описват връзките между растенията, тревопасните и месоядните животни.

$$p' = sun\_bio – 4.03 * p$$

$$h' = 0.48 * p – 17.87 * h$$

$$c' = 4.85 * h – 4.65 * c$$

Структура на основните компоненти

Първото равенство означава, че има експоненциално намаляване на биомасата на растенията, както и увеличение породено от слънчевата светлина, което кара растенията да растат.

Подобно второто равенство описва скоростта на промяна на биомасата на тревопасните. Намалението на биомасата е възпрепятствано от увеличението на консумирането (унищожаването) на растенията. Поведението на месоядните е аналогично.

Структура на основните компоненти

Загубата на биомаса в околната среда и мъртвата органична материя се дава с:

$o' = 2.55 * p + 6.12 * h + 1.95 * c$

$e' = 1.00 * p + 6.90 * h + 2.70 * c$

Двете равенства дават скоростта на:

- промяната на биомасата, която се съдържа в мъртвия органичен материал;
- губещата се биомаса в околната среда.

O и *e* са пропорционални на биомасата на растенията, тревопасните и месоядните животни.

Структура на основните компоненти

Променливата *T* представя симулационното време.
Прогресът на симулационно време се контролира автоматично от системата за контрол на изпълнението на симулационната среда.
T не е задължително да се дава от потребителя.
Не е задължително да се декларира *T*.

Моделно описание за компонента CedarBog.

```
BASIC COMPONENT CedarBog

USE OF UNITS
TIMEUNIT = [a]

DECLARATION OF ELEMENTS
   CONSTANTS
     Pi       (REAL)     := 3.14,
     Bio_Fac  (REAL[a])  := 1E-15 [a]

   STATE VARIABLES
   CONTINUOUS
     p (REAL[t]) := 0 [t],
     h (REAL[t]) := 0 [t],
     c (REAL[t]) := 0 [t],
     o (REAL[t]) := 0 [t],
     e (REAL[t]) := 0 [t]

   DEPENDENT VARIABLES
   CONTINUOUS
     sun     (REAL[kJ/m^2]) := 0  [kJ/m^2],
     sun_bio (REAL[t/a])    := 0  [t/a]

DYNAMIC BEHAVIOUR
   sun := 95.9 [kJ/m^2] *(1 + 0.635 * SIN (2[1/a]*Pi*T));
   sun_bio := sun * Bio_Fac;

   DIFFERENTIAL EQUATIONS
     p' := sun_bio       -  4.03[1/a] * p;
     h' := 0.48[1/a] * p - 17.87[1/a] * h;
     c' := 4.85[1/a] * h -  4.65[1/a] * c;
     o' := 2.55[1/a] * p +  6.12[1/a] * h + 1.95[1/a] * c;
     e' := 1.00[1/a] * p +  6.90[1/a] * h + 2.70[1/a] * c;
   END

END OF CedarBog
```

Структура на основните компоненти

Преди симулационното изпълнение Run1 се дават стойности на контролните параметри. На фигурата са показани подходящи стойности за модела MCedarBog. Промените се правят чрез избиране на обект Control parameters и викане на командата Configure control parameters. След въвеждане на параметрите, трябва да се извика функцията Check.

Структура на основните компоненти

Създадат се наблюдатели с цел в тях да се запишат и представят резултатите на избраните динамични редове. В нашия случай, променливите са Слънце (sun), Растения (p), Месоядни (c) и Органични (o).

Структура на основните компоненти

Всички моделни променливи трябва да имат начални стойности. Симулационното изпълнение продължава до T = 2.0. Динамичните редове, които са записани в каталога Simulations results, могат да бъдат показани и анализирани. Подкаталога Statistics в директория Protocols осигурява информация за напредъка на симулационното изпълнение - например броя на стъпките или средната стойност на дължината на стъпката.

Структура на основните компоненти

Синтаксис за описване на основни компоненти.

Синтаксиса за описване на основните компоненти е:

```
basic_component : : = BASIC COMPONENT identifier
                       [mobile_subclass_declaration]
                       [unit_defenition_part]
                       [local_definitions]
                       declaration_of_elements
                       [dynamic_behaviour]
                       END OF identifier
```

Една основна компонента се състои от име (Identifier), декларационна част и описание на динамичното поведение.

Структура на основните компоненти

Квадратните скоби посочват, че оградената секция не се нуждае да бъде представяна. Редът на секциите трябва да се спазва:

mobile_subclass_declaration

Тук се декларират класове за подвижни компоненти, които се използват в основните компоненти.

unit_definition_part

Структура на основните компоненти

На променливите на модела и числата могат да се посочат мерни единици.

local _definitions

Секцията съдържа изброени променливи, списък функции и списък на случайни разпределения.

declaration_of _elements

Тук се декларират константи, променливи, произволни променливи, индикатори и региони, използвани в модела.

Структура на основните компоненти

Ключовата дума BASIC COMPONENTS е последвана от името на компонентата.

Декларационната част съдържа само declaration_of_elements.

Синтаксисът на декларацията на елементите е:

declaration_of_elements :: = DECLARATION OF ELEMENTS
[list_of_constants]
[list_of_state_variables]
[list_of_dependent_variables]
[list_of_sensor_variables]
[list_of_random_variables]
[list_of_transition_variables]
[list_of_sensor_indications]
[list_of_locations]
[list_of_sensor_locations]

Структура на основните компоненти

Тези елементи могат да се разделят в четири групи:

- Променливи на модела;
- Случайни променливи;
- Индикатори;
- Региони.

Променливите на модела описват атрибутите, които всяка компонента притежава. Всяка променлива на модела изисква:

- Употреба;
- Поведение като функция на времето;
- Тип.

Променливи на състоянието

Използват се за описание на динамични свойства на компонента. Тяхната първа производна относно времето може да се използва в диференциалните уравнения. Техните стойности могат да бъдат модифицирани от дискретни събития.

Структура на основните компоненти

Зависима променлива:
Зависимите променливи се дефинират чрез уравнения. Дясната част на алгебрично уравнение може да съдържа променливи на състоянието, други зависими променливи или времето.

Сензорна променлива:
Сензорните променливи са свързани с променлива в друга компонента, използвайки връзка в компонента от по-високо равнище. Сензорната променлива позволява достъп до променливи от други компоненти. Може да не се дават инициализационни стойности на сензорните променливи.

Структура на основните компоненти

Времево поведение на моделните променливи:

- дискретно

Стойността на тези променливи може да се промени от отделни събития;

- непрекъснато:

Тези променливи се променят и по непрекъснат и по дискретен начин. Непрекъснатите променливи се описват от диференциални уравнения.

Типове променливи на модела:

- цял;
- реален;
- логически;
- изброим.

Структура на основните компоненти

Моделът *CedarBog* не съдържа никакви *submit_declaration* или *local_definitions*. Нужно е само *declaration_of_elements*.

Списъкът от елементи се състои от константи, променливи на състоянието и зависими променливи.

Времевото поведение трябва да се определи за всички променливи на състоянието и зависими променливи. В този случай и двете са непрекъснати.

Всички моделни променливи трябва да получат тип. Тук всички са от тип *Real*.

Структура на основните компоненти

Синтаксиса на динамичното поведение има следната форма:

dynamic_behaviour :: = DYNAMIC BEHAVIOUR statement_sequance

statement_sequance :: = statement {statement}

Фигурните скоби означават, че съдържанието им може да е празно, или че те могат да се повтарят неограничен брой пъти.

statement :: = algebraic_equation
| differential_equations
| region_defining_statement
| event_defining_statement

Структура на основните компоненти

Реда на запис на алгебричните уравнения, диференциалните уравнения и събитията е произволен.

- Алгебрични уравнения:

Алгебричните уравнения са дефиниращи уравнения за зависимите променливи от лявата страна на ‘:=’;

- Диференциални уравнения:

Позволени са диференциални уравнения от първи ред;

- Събития:

Събитията описват промените на променливите на състоянията. Стартирането на събитие се случва, когато състоянието на модела отговаря на определените в събитието условия.

Структура на основните компоненти

Моделът *CedarBog* съдържа алгебрични уравнения за зависимите променливи *sun* и *sun_bio*.
Забележка:
SIMPLEX MDL е декларативен и непроцедурен когато става дума за описание на динамика.
След ключовата дума DIFFERENTIAL EQUATIONS се записват диференциалните уравнения за непрекъснатите променливи *p*, *h*, *c*, *o* и *e*.

Структура на основните компоненти

Забележки:

- Ключовите думи са част от синтаксиса. Те винаги се записват с главни букви в SIMPLEX MDL. Елементите на декларацията са разделени чрез запетайки, освен за последния. Изразите завършват винаги с точка и запетая.

Примерния модел CedarBog е непрекъснат.

Алгебрични уравнения

Алгебричните уравнения в SIMPLEX MDL дефинират зависима променлива. Дясната страна на дефиниращото уравнение може да съдържа променливи на състоянието, зависими променливи или време.

Алгебрични уравнения

Пример. Декларирани са две зависими променливи A и B.

DEPENDENT VARIABLES

DISCRETE

A (REAL) : = 0,

B (REAL) : = 0

Нека те са дефинирани от следните две алгебрични уравнения:

DYNAMIC BEHAVIOUR

A : = 2;

B : = A + 1;

Алгебрични уравнения

От тези дефиниции, тези зависими променливи винаги имат следните стойности:

A = 2

B = 3

Независимост на реда означава, че алгебричните равенства могат да се напишат така:

B : = A + 1;

A : = 2;

Трябва да е вярно все още и:

A : = 2 B : = 3

Ясно е, че алгебричните уравнения не трябва да се бъркат с изразите в процедурните езици за програмиране.

Алгебрични уравнения

За да се уверим, че алгебричните уравнения винаги дават същите резултати, независимо от реда им, те се оценяват итеративно (последователно) докато стойностите на зависимите променливи повече не се различават от тези на предишната итерация.

Първо показваме какво се случва, когато равенствата са написани в следния ред:

A : = 2;

B : = A + 1;

Алгебрични уравнения

Нови стойности се изчисляват от старите. След второто изчисляване на алгебричните уравнения, стойностите на A и B нямат промяна и стойностите

A = 2   B = 3

се взимат като финални.

Бяха нужни две итерации. Последната итерация се използва само за проверка дали итерацията на алгебричните уравнения може да се прекъсне.

Ако алгебричните уравнения се напишат в следния ред:

B : = A + 1;   A : = 2;

Алгебрични уравнения

Сега са необходими три итерации за изчисляването.

Максималния брой итерации може да се определи от потребителя, използвайки контролния параметър *MaxCyc*, който може да се промени в каталога *Control parameters* във всяко изпълнение чрез извикване на командата *Configure control parameters*. Подходящото поле във входния прозорец може да се открие в горната лява страна на секцията *Algebraic equations* под името *Max.#Cycles*. Той има стойност по подразбиране 50.

Итерацията прекъсва, когато стойностите на зависимите променливи не се променят с повече от *EPS*. Стойността на *EPS* може да се промени в полето *Epsilon*. То има стойност по подразбиране 1E–10.

Алгебрични уравнения

Забележки:

- Тъй като алгебричните уравнения могат да се оценят много често, препоръчително е те да са подредени подходящо в описанието на модела. Това може да спести времето на изчисление;
- Независимостта на реда от алгебрични уравнения е необходима, за да е възможно да се раздели една компонента на подкомпоненти по произволен начин. Независимостта на реда гарантира резултатите да са идентични, независими от реда, в който подкомпонентите са извикани.

Алгебрични уравнения

Възможно е следното разширение на модела Biotope_1. Представено е ограничение на пашата, което осигурява, че броят на зайците не превишава MaxCap = 1500. Вместо

Hare`= a * Hare

сега е налице:

Hare` = (MaxCap – Hare) / MaxCap * a * Hare

Скоростта на нарастване за зайците стига до нула когато броят на зайците приближава MaxCap.

Алгебрични уравнения

На първа итерация получаваме:

Hare_Inc = 0 Cap = 0.733

Fox_Dec = -46.25 Hits = 555

На втора итерация получаваме:

Hare_Inc = 513.33 Cap = 0.733

Fox_Dec = -46.25 Hits = 555

Simpex3 може да изведе детайлен протокол на оценката на алгебричните уравнения и преходите между състоянията. За да се активира тази функция се активира бутона Options в долната дясна част на прозореца на контролните параметри в секция Protocol.

В протокола са се отпечатали само тези стойност, които са се изменили от последната ситуация. Ако дадена стойност не присъства в протокола това означава, че тя остава непроменена от предходния цикъл.

Алгебрични уравнения

```
 BASIC COMPONENT Biotope_3
  USE OF UNITS
     UNIT[NumH] = BASIS
     UNIT[NumF] = BASIS
     TIMEUNIT = [a]
  DECLARATION OF ELEMENTS
     CONSTANTS
    a (REAL[1/a])  :=  1.75 [1/a], # Скорост на нарастване на зайците
    b (REAL[1/a])  := -1.25 [1/a], # Скорост на нарастване на лисиците
    c (REAL[1/(NumH*NumF*a)])            # Вероятност на срещите
      := 0.0375 [1/(NumH*NumF*a)],         #
MaxCap(INTEGER[NumH]):= 1500[NumH],# Максимален капацитет
      BFacH(REAL[NumH]) := 1.0[NumH],     # Коефициент на плячка
зайци
      BFacF(REAL[NumF]) := 0.1[NumF]       # Коефициент на плячка
лисици
```

Алгебрични уравнения

```
STATE VARIABLES
  CONTINUOUS
    Hare  (REAL[NumH]) := 400[NumH],
    Fox   (REAL[NumF]) :=  37[NumF]

  DEPENDENT VARIABLES
  CONTINUOUS
    Cap       (REAL),                 # Свободен капацитет
    Hare_Inc  (REAL[NumH/a]),         # Увеличение на зайците
    Fox_Dec   (REAL[NumF/a]),         # Намаляване на лисиците
    Hits      (REAL[1/a])             # Срещи
```

Алгебрични уравнения

```
DYNAMIC BEHAVIOUR

   Hare_Inc  := Cap * a * Hare;
   Cap       := (MaxCap - Hare) / MaxCap;
   Fox_Dec   := b * Fox;
   Hits      := c * Hare * Fox;

   DIFFERENTIAL EQUATIONS
     Hare' :=  Hare_Inc - BFacH * Hits;
     Fox'  :=  Fox_Dec  + BFacF * Hits;
   END
```

Алгебрични уравнения

```
 DYNAMIC BEHAVIOUR
   Cap        := (MaxCap - Hare) / MaxCap;
Hare_Inc  := Cap * a * Hare;
Hits      := c * Hare * Fox;
Hare' :=  Hare_Inc - BFacH * Hits;
END
Fox_Dec   := b * Fox;
DIFFERENTIAL EQUATIONS
      Fox'  :=  Fox_Dec  + BFacF * Hits;
   END
```

Алгебрични уравнения

Забележка:
Позволени са и алгебричните уравнения от следния вид:
y = y – 0.5 * (y – (x/y))
Стойността на *y* се оценява итеративно докато или максималния брой итерации е достигнат, или разликата между старите и новите стойности на *y* са станали по-малки от EPS. Това позволява формиране на рекурентни алгебрични уравнения. Позволено е алгебричното уравнение A = A + 1. Уравнението A : = A + 1 се изчислява многократно, докато се достигне MaxCyc, увеличавайки стойността на A с 1 всеки път.

Събития

Едно събитие причинява промяна в състоянието за променливите на състоянията в дискретни точки от време.

Събитието се състои от следните три части:

- Стартиращ механизъм;
- Процедурна част; (незадължителна)
- Описанието на промените на състоянието.

Синтаксиса има формата:

```
event_defining_statement :: = triggering_mechanism
                              [procedural_part]
                              transitions_part
```

Събития

Има две възможности за стартиращия механизъм:

- Удовлетворени стойности на условие;
- Чрез индикатор.

Синтаксиса на стартиращия механизъм е:

triggering_mechanism :: WHENEVER expression

/ON indication {OR indication}

Събития

Стартиращия механизъм и конструкцията WHENEVER.

Конструкцията WHENEVER причинява случването на събитие, когато логическия израз след нея има стойност TRUE.

Това означава, че определено състояние на модела, което е представено от логическия израз, причинява промяната в състоянието на модела.

Важно е да се отбележи, че промените на състоянието в събитие, определено от WHENEVER конструкция, се изпълняват докато стойността на логичния израз е TRUE. Промените на състоянието трябва да покажат, че истинната стойност се променя на FALSE, за да се избегне безкраен цикъл.

Обратно, ON конструкцията причинява събитие да се случи само, когато истинната стойност на условие се промени от FALSE на TRUE. Ако стойността остане TRUE, събитието не се пуска от ON конструкцията.

Събития

Събитията се изпълняват поради стартиращия механизъм.
В допълнение към незадължителната процедурна част, събитието съдържа описание на промените на състоянието.
Това описание включва:
• Дефиниране на промяната на състоянието;
• Сигнални команди;
• Дефиниране на региони за събития;
• Команди за визуализация.

Събития

Синтаксисът е следния:

transition_part ::= DO [TRANSITION[S]
transition_statement_sequence
END

transition_statement_sequence ::= transition_statement
{ transition_statement }

transition_statement ::= state_transition_definition
| signal_satement
| event_region_defining_statement
| display_statement

state_transition_definition ::= state_variable_assinment
transfer_statement

state_variable_assinment ::= selected_element
'^' ':=' ex[ression ';'

Събития

Избираме като илюстрация модела CedarBig. Моделът се разширява от едно събитие. Мъртвата органична материя на дъното на езерото надхвърля o = 30[t].

Като резултат от това събитие, стойността на променливата на състоянието е поставена да е o = 0 [t] и събитието не се изпълнява.

Събития

Пример. Моделът Empty с едно събитие.

```
BASIC COMPONENT Empty

  USE OF UNITS
    TIMEUNIT = [a]

  DECLARATION OF ELEMENTS

    CONSTANTS
      Pi      (REAL)    := 3.14,
      Bio_Fac (REAL[a]) := 1E-15 [a]
```

Събития

```
STATE VARIABLES
  CONTINUOUS
   p (REAL[t]) := 0 [t],        # Растения
   h (REAL[t]) := 0 [t],        # Тревопасни
   c (REAL[t]) := 0 [t],        # Месоядни
   o (REAL[t]) := 0 [t],        # Запас
   e (REAL[t]) := 0 [t]         # Загуба на среда

 DEPENDENT VARIABLES
 CONTINUOUS
   sun     (REAL[kJ/m^2]) := 0 [kJ/m^2],   # Solar radiation
   sun_bio (REAL[t/a])    := 0 [t/a]

DYNAMIC BEHAVIOUR
   sun := 95.9 [kJ/m^2] *(1+0.635 * SIN (2[1/a] * Pi *T));
   sun_bio := sun * Bio_Fac;
```

Събития

```
DIFFERENTIAL EQUATIONS
    p' := sun_bio - 4.03[1/a] * p;
    h' := 0.48[1/a] * p - 17.87[1/a] * h;
    c' := 4.85[1/a] * h -  4.65[1/a] * c;
    o' := 2.55[1/a] * p +  6.12[1/a] * h + 1.95[1/a] * c;
    e' := 1.00[1/a] * p +  6.90[1/a] * h + 2.70[1/a] * c;
  END

  WHENEVER o > 30 [t]
  DO
   o^ := 0 [t];
  END
END OF Empty
```

Събития

Пример. Еднократно изпълнение на събитие.

```
BASIC COMPONENT PrintSun

USE OF UNITS
   TIMEUNIT = [a]

DECLARATION OF ELEMENTS

   CONSTANTS
     Pi      (REAL)    := 3.14,
     Bio_Fac (REAL[a]) := 1E-15 [a]
```

Събития

```
STATE VARIABLES

  DISCRETE
    Printed (LOGICAL) := FALSE

  CONTINUOUS
    p (REAL[t]) := 0 [t],        # Plants
    h (REAL[t]) := 0 [t],        # Herbivores
    c (REAL[t]) := 0 [t],        # Carnivores
    o (REAL[t]) := 0 [t],        # Deposits
    e (REAL[t]) := 0 [t]         # Loss to environment
```

Събития

```
DEPENDENT VARIABLES
  CONTINUOUS
    sun     (REAL[kJ/m^2]) := 0 [kJ/m^2],   # Solar radiation
    sun_bio (REAL[t/a])    := 0 [t/a]

DYNAMIC BEHAVIOUR

    sun := 95.9 [kJ/m^2] *(1+0.635 * SIN (2[1/a] * Pi *T));
    sun_bio := sun * Bio_Fac;
```

Събития

```
 DIFFERENTIAL EQUATIONS
     p' := sun_bio - 4.03[1/a] * p;
     h' := 0.48[1/a] * p - 17.87[1/a] * h;
     c' := 4.85[1/a] * h -  4.65[1/a] * c;
     o' := 2.55[1/a] * p +  6.12[1/a] * h + 1.95[1/a] * c;
     e' := 1.00[1/a] * p +  6.90[1/a] * h + 2.70[1/a] * c;
  END
WHENEVER (sun < 95.9 [kJ/m^2]) AND NOT (Printed)
DO
  DISPLAY("T= %f Radiation falls below %f \n", T, sun);
  Printed^ := TRUE;
END
WHENEVER (sun >= 95.9 [kJ/m^2]) AND (Printed)
DO
  Printed^ := FALSE;
END
END OF PrintSun
```

Събития

Позволени са сравнения в логически израз за конструкцията WHENEVER.

Допустими са операции:

„=” : равно;

„<>” : различно;

„<” : по-малко от;

„<=” : по-малко или равно;

„>” : по-голямо от;

„>=” : по-голямо или равно.

Събития

Логическия израз за WHENEVER конструкцията може се съдържа времето *T*. *T* е от тип REAL, стойността му винаги е достъпна и не се нуждае от декларация.

Събития

Примери:

Ако събитието се регистрира, когато симулационния часовник *T* достигне стойността 10, тогава трябва да се използват следните условия :

WHENEVER T >= 10;

Моделът Empty може да бъде разширен. В модела залагаме, че преди T = 1.0 не може да настъпи изследваното събитие. Логическото условие има следната форма:

WHENEVER (T >= 1.0 [a]) AND (o > 30 [t])

Като следствие на това ограничение, *o* първоначално ще расте. В момента T = 1 [a], стойността се нулира o = 0 [t], докато в този момент време „o” е вече значително по-голямо от 30 [t]. Оттук нататък, намалението се появява всеки път, когато „o” достигне стойността 30 [t], докато T > 1.0 [a].

Събития

Стартиращ механизъм и ON конструкцията

Всички условни крайни промени на състояния могат да се опишат, използвайки WHENEVER конструкция. ON конструкцията е полезно допълнение. То позволява едно събитие да бъде стартирано само веднъж при дадена индикация.

Синтаксиса има следния вид:

```
triggering_mechanism :: = WHENEVER expression
                          |ON indication {OR indication}
indication :: = indexed_identifier
                |START
                |”^” expression “^”
```

Събития

Примери:

В модела Empty стойността за *o* е дефинирана като *o*=0 [t], винаги когато е изпълнено условието *o*>30 [t]. Тази процедура може да бъде записана използвайки конструкцията WHENEVER, тъй като чрез установяването на *o* като *o*=0 [t], условието става лъжа и събитието не се изпълнява втори път. Възможно е да се замени WHENEVER конструкцията с ON конструкцията.

Това води до следния запис:

```
ON ^o>30 [t]^
DO
  o^:=o [t];
END
```

Събития

Ако например, условието o>30 [t] води до еднократно действие, което намалява броя на растенията с 10%. Описанието трябва да е следното:

```
ON ^o>30 [t]^
DO
   p^:=p-p/10;
END
```

Събития

В моделът PaintSun, извеждането на слънчевата енергия е много по-просто с използването на ON-конструкцията. Логическата променлива Printed повече не се нуждае от анулиране. Когато радиацията падне под 95.9 [kJ/m^2], събитието се изпълнява само веднъж. Описанието има следния вид:

```
ON ^sun<95.9 [kJ /m ^ 2]^
DO
   DISPLAY (“ T = % f Енергията пада под % f \n”, T, sun);
END
```

Събития

Индикаторите се декларират в декларационната част:

Пример:

```
DECLARATION OF ELEMENTS
STATE VARIABLE
    A (INTEGER) : = 0
TRANSITION INDICATORS
Indic
DYNAMIC BEHAVIOUR
ON ^ T > = 10 ^
DO SIGNAL Indic;
END
ON Indic
DO A ^ : = 1;
END
```

Събития

В момента T = 10, е поставен индикатора Indic.

Възможна е употребата на стандартен указател START.

Пример:

```
ON START
DO
  A^:=0;
END
```

В този случай, първо се обработва събитие при започването на симулационното изпълнение.

START индикатора е необходим, защото промяна в логическата стойност на условие не може да се установи, докато няма предишна стойност.

Забележка:

Индикатора START е стандартен. Не е необходимо да се декларира.

Събития

Преходи в събитие.

Събитията се състоят от стартиращ механизъм, процедурната част и преходи. Възможни са следните конструкции за преходите:

- Промени на състояние;
- Командата SIGNAL;
- Разделение на пространството на състояния от събитие;
- Командата DISPLAY.

Синтаксиса е:

```
transition_statement :: = state_transition_definition
                          |signal_statement
                          |event_region_defining_statement
                          |display_statement
```

Събития

Частта state_transition_definition описва възможностите за промени на състояние.

Има две възможности за определяне на промяната на състоянието:

- Промени на състоянието за променливи на състоянието;
- Преместващи команди за подвижни компоненти.

Преходите между състояния при събития са позволени за непрекъснати и за дискретни променливи на състоянията.

Забележка:

Зависима променлива, чиято стойност се изменя само от събития, трябва да се декларира като дискретна. Непрекъснатите променливи на състоянията могат да бъдат променени от събитие.

Събития

Синтаксисът за промени на състояния за дискретни или непрекъснати променливи на състоянията има следната форма:

state_transition_definition :: = state_variable_assignment
| transfer_statement

state_variable_assignment :: = selected_element ‘^’ ‘: =’ expression ‘;’

Събития

Примери:

- Моделът Empty съдържа следната променлива на състоянието:

o ^ : = 0 [t];

Променливата „o” е непрекъсната. Тя може да се променя от събитие;

- В моделът PrintSun е поставена дискретната променлива Printed. Printed е обявена като дискретна променлива на състоянието от тип LOGICAL, например:

Printed ^ : = TRUE;

Следващата възможност за промяна на състояние е командата SIGNAL. Тя поставя индикатор в събитие.

Командата SIGNAL се прилага към потребителски дефиниран индикатор. Стандартните индикатори START и STOP са на разположение по подразбиране и не е нужно да бъдат декларирани. Последният кара симулацията да спре.

Това позволява формулировката на произволен стоп-критерий.

Събития

Пример:

Моделът CedarBog се изпълнява до T = 2 [a].

По-нататък стоп критерий може да бъде, когато залежите (отлаганията) достигнат стойност o = 150 [t]. Допълнителното събитие ще изглежда така:

```
ON      ^o > 150 [t]^
DO
   SIGNAL STOP;
END
```

Синтаксисът на командата SIGNAL има следния вид:

```
signal_statement :: = SIGNAL indexed_identifier ‘;’
                      | SIGNAL STOP ‘;’
```

Събития

По-нататъшна възможност е да се раздели пространството на състоянията от събитие. Това осигурява възможността за специфициране на различни преходи, зависещи от състоянието на модела.

Синтаксисът е:

```
event_region_defining_statement ::= IF expression
                                    DO transition_statement_sequence END
                                     { ELSEIF expression
                                     DO transition_statement_sequence END }
                                     { ELSE expression
                                     DO transition_statement_sequence END }
```

Пример:

Разглеждаме модела CedarBog. Чрез извършване на действия в интервали от 0.2 [a], количеството на растенията се запазва между 22 [t] и 26 [t]. Това изисква събитие, което се стартира редовно.

Промяната на състояние зависи от състоянието на модела в съответния момент от време.

Събития

Условие	Действие
p < 20[t]	20% добавени растения
20[t] < p < 22[t]	10% добавени растения
22[t] < p < 26[t]	Растения без непромяна
26[t] < p < 28[t]	10% растения премахнати
p > 28[t]	20 % растения премахнати

Събития

Събитието има следната форма:

```
WHENEVER T >= Tstep
DO
   TSTEP^ := T +0.2;
   IF  p < 20 [t]
                 DO p^:= p + p/5; END
   ELSEIF   (p > = 20 [t]) AND (p < 22 [t])
                 DO p^:= p + p/10; END
   ELSEIF   (p > = 26 [t]) AND (p < 28 [t])
                 DO p^:= p - p/10; END
ELSEIF p > = 28 [t]
    DO    p^ := p – p/5;END
END
```

Забележки:

Възможно е разделянето на пространство на състоянията при събитие. Ако събитие е стартирано, преходите са зависими от състоянието на модела.

Съществува възможността за специфицирането на различни динамики, основани на състоянието на модела.

Събития

Синтаксиса е:

display_statement ::=DISPLAY ‘(‘ string {‘,’expression}’)’ ‘;’

string :: = ‘ ” ’ { ascii_character} ‘ ” ’

Символът на низ съответства на този на printf командата в езика *C*. Броят и типът на изразите, които трябва да съответства на низа.

Бележка:

Тук не се изпълнява семантична проверка. Неправилното използване на командата може да доведе до грешка по време на изпълнението като segmentation violation, bus error или floating point overflow.

Събития

За различните типове се използват следните формати:

INTEGER : % ld

REAL : %f

LOGICAL : %d

Enumerated type : %d

Time T : %f

Стойностите на променливи от логически тип и изброими типове могат само да се изведат чрез тяхното вътрешно представяне. Те съответстват на цели числа в описанието на декларацията.

Специални символи:

\n Нов ред

\f Нова страница

Събития

Примери:

DISPLAY (“ Arrival of a customer \n ”);

DISPLAY (“ Arrival at time T = %f \n ”, T);

DISPLAY (“ I = %ld   X = %f   L = %d   E= %d \n”,
I, X, L, E);

I:INTEGER, X: REAL

L:LOGICAL

E: Enumerated type

DISPLAY (“ Area = %f \n ”, X*Y);

Събития

Процедурна част на събитие.

Събитията може да съдържат възможна процедурна част. Тя позволява алгоритми да се включват в моделното описание. Процедурната част се състои от декларативна част и част с изразите. Резултатите се съхраняват във временни променливи и се пренасят в декларативната секция на събитията. Временните променливи не са дефинирани извън събитието. Когато събитието се изпълни втори път, те се инициализират отново. Ако декларацията им не им дава начална стойност, те се инициализира с 0 или FALSE.

Събития

Процедурната секция може да съдържа:

- Присвоявания;
- Условия;
- Цикли;
- Команди за изход;
- Команди за визуализация.

Събития

Синтаксис:

procedural_assinment ::= indexed_identifier ‘:=’ expression ‘;’

Пример:

```
ON^T >= TNext^
DECLARE
    X(REAL)            # Дължина на вектора (x1,x2)
DO PROCEDURE
   X := SQRT (x1*x1 +x2*x2);
END
DO TRANSITIONS
 Y1^ := x1/x;    # Присвояване на променливите на състоянието
 Y2^ := x2/x;    # Формира се единичния вектор (y1,y2)
END
```

Събития

Упражнения:

Представете стоп критерий в модела CedarBog. Симулацията трябва да се изпълнява поне докато T = 20 [a]. Трябва да приключи след T = 20 [a], когато растенията са със стойност p = 30.0 [t].

Условието за спиране е:

(T > = 20 [t]) AND (p > = 30.0 [t])

Събития

Езерото се освобождава от залежи на всеки 0.5 времеви единици.

Упътване:

Въвежда се нова променлива на състоянието TNext, която отбелязва кога се случва събитието. Допълнителното събитие има вида:

WHEREVER T > = TNext

DO

o^ := 0 [t]

TNext^ := TNext + 0.5 [a] ;

END

Събития

Езеро се чисти от залежите на всеки 0.5 единици време. Това се прави само, ако е нужно, т.е. когато o > 50 [t].

Упътване:

Събитието има вида:

```
WHENEVER   (T > = TNext) AND (o > 50 [t])
DO
        o^ := 0 [t]
        TNext^ := TNext + 0.5 [a];
END
```

Събития

Нарастване на времето в модели с крайно време.

В Simplex3 стартиращата контролна част на изпълняваната система е отговорен за управлението на преходите между различните състояния.

В SIMPLEX MDL едно състояние винаги се характеризира от време и цикъл.

По всяко време и при всеки цикъл променливата на състоянието е в точно едно състояние.

z (tn, ki)   Състояние

tn   Време

ki   Цикъл

Зависим от времето преход в състоянието:

z (tn + 1, k1) = ft(z(tn, ki))

Преход в условно състояние:

Z (tn, ki + 1) = fk(z(tn, ki))

Преход в състояние, зависещо от времето е налице когато стойността на променлива на състоянието се промени в дискретен момент време.

Събития

Промени на цвета на светофар в зависимост от времето:

Време	Състояние
T = 0	Червено
T = 5	жълто
T = 6	зелено
T = 11	жълто
T = 12	червено
и т.н.	

Събития

С начална стойност TNext=0 и състояние жълт, събитието ще има следната форма:

WHENEVER T>=TNext

DO

State^ := ‘red’;

TNext := 12;

END

Времето T ще присъства винаги в събитие, зависещо от времето.

Събития

Събитието води до ново състояние на модела, което се достига при цикъл 1. Фактът, че новото състояние на модела не става валидно до цикъл 1 е отбелязан със символа ‘^’.

В началото на симулацията, при цикъл 0, имаме начални стойности, които се изменят от зависещо от времето събитие при цикъл 1, получавайки ново състояние за модела.

Събития

Също при събитие променливата TNext получава нова стойност. Това осигурява, че условието на събитието още веднъж ще стане истина при T = 12, така че събитието отново ще се регистрира.

Условните събития се появяват, когато при специален цикъл, състоянието на модела дава възможност за по-нататъшни събития, които водят до ново състояние на модела при следващия цикъл.

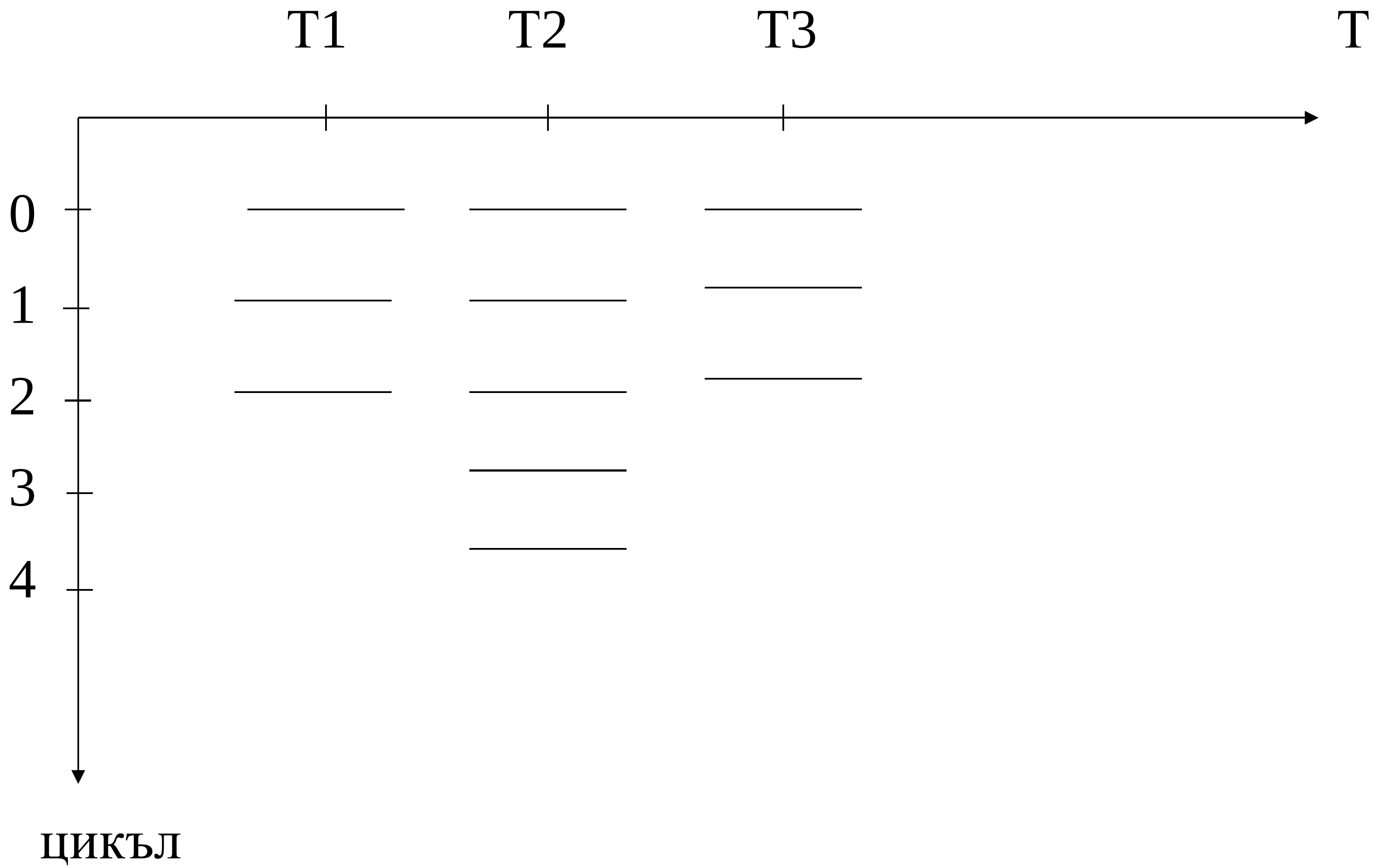
Събития

В този случай, условното събитие важи за състоянието на модела при цикъл 1 и го променя в ново състояние в цикъл 2.

Възможно е състоянието на модела при специално време и цикъл да дава възможност на повече от едно събитие. Всички тези събития започват от оригиналното непроменено състояние. Когато всички събития се изпълнят, цикълът напредва и новото състояние става валидно.

Събития

Цикли и времеви стъпки за време T1 – състояния в цикли 0, 1 и 2.
При цикъл 0 имаме начално състояние.



Събития

Състоянията водят до цикъл 1. Новото, изменено състояние на модела активира по-нататъшни събития, които водят до състоянието в цикъл 2.

Ако не са възможни повече условни събития, последователността от цикли свършва и симулационния часовник напредва поради следващото зависимо от времето събитие.

Събития

Моделът брояч.

Частици пристигат при един Гайгеров брояч с експоненциално разпределение със среден интервал от пет времеви единици. Всеки път, когато се открие частица, броячът се увеличава с единица. Броячът на откритите частици се отпечатва когато е кратен на десет.

Събития

```
BASIC COMPONENT Counter

DECLARATION OF ELEMENTS

   STATE VARIABLES
     Count (INTEGER) := 0,
     TNext   (REAL)    := 0,
     NPrint  (INTEGER) := 0

   RANDOM VARIABLES
     Interval (REAL) : EXPO
    (Mean:=5,LowLimit:=0.15,UpLimit:=25)
```

Събития

```
DYNAMIC BEHAVIOUR

# Събитие 1
WHENEVER T >= TNext
DO
   Count^   := Count + 1;
   TNext^   := T + Interval;
END

# Събитие 2
ON ^IMOD(Count, 10) = 0^
DO
   DISPLAY ("Counter = %d \n", Count);
   NPrint^ := NPrint + 1;
END

END OF Counter
```

Събития

При време T = 0 и cycle = 0 се удовлетворяват началните стойности. Докато симулационния часовник T е равен на TNext, действа събитието. Времето на следващия момент TNext е дефинирано в събитието от генератора на случайни числа, който доставя експоненциално разпределени случайни числа.

В момента T = 0 и cycle = 1 променливите на състоянието имат следните стойности:

Counter = 1

TNext = TRandom

Събития

Докато не се изпълни указанието за събитие 2, то събитие 2 не се случва. За време T = 0 са налице само циклите 0 и 1. Докато при време не са възможни по-нататъшни събития, симулационния часовник напредва към следващото време, при което се очаква събитие, зависещо от времето. Това е времето съхранявано в променливата TNext.

След като пристигне десетата частица се активира второ събитие. Започвайки от нова, увеличена стойност на брояча при цикъл 1, извеждането може да се извърши. В цикъл 2 се съдържа новото състояние на модела с увеличена стойност на NPrint.

Събития

Моделът тест.

Моделът Test съдържа три дискретни променливи на състоянията A, B и C и променлива на състоянието TNext за нарастването на времето.

События

```
BASIC COMPONENT Test

DECLARATION OF ELEMENTS

  STATE VARIABLES
  DISCRETE
    A     (INTEGER) := 0,
    B     (INTEGER) := 0,
    C     (INTEGER) := 0,
    TNext (INTEGER) := 0

  TRANSITION INDICATORS
    End
```

Събития

```
DYNAMIC BEHAVIOUR


# Събитие 1
WHENEVER T >= TNext
DO
   DISPLAY ("A= %d, B= %d, C= %d \n", A, B, C);
   TNext^ := TNext + 1;
   A^ := 2;
END
```

Събития

```
# Събитие 2
ON ^(A = 2)^
DO
   DISPLAY ("A= %d, B= %d, C= %d \n", A, B,
   C);
   A^ := 1;
   C^ := 1;
END
# <- Цикли и последващи събития
```

Събития

```
# Събитие 3
ON ^(A = 2) AND (B = 0)^
DO
   DISPLAY ("A= %d, B= %d, C= %d \n", A,
B, C);
   B^ := A;
END
```

Събития

```
# Събитие 4
ON ^(B = 2)^
DO
   DISPLAY ("A= %d, B= %d, C= %d \n", A, B,
   C);
   A^ := 0;
   B^ := 0;
   C^ := 0;
   SIGNAL End;
END
```

Събития

```
# Събитие 5
ON End DO
   DISPLAY ("A= %d, B= %d, C= %d \n", A, B, C);
END

END OF Test
# <- Цикли и последващи събития
```

Събития

В началото на симулационното изпълнение при T=0 имаме следното началното състояние:

Takt0			
A	B	C	TNext
0	0	0	0

Събития

Ако сега проверим петте възможни събития ще видим, че стартиращия механизъм за събитие 1 ще позволи събитие 1 да се осъществи. Това събитие води до ново състояние на модела. Това ново състояние на модела принадлежи на цикъл 1.

Събития

След първото събитие моделът е в следното състояние:

Takt1			
A	B	C	TNext
2	0	0	1

Събития

Проверка на събитията показва, че първото събитие е активирало последващите събития 2 и 3. И двете събития започват от състоянието на модела в цикъл 1. Промените на съставните променливи не стават действителни до следващия цикъл.

Събития

Състоянието на модела при цикъл 2 е:

Takt2			
A	B	C	TNext
1	2	1	1

Събития

Това състояние при цикъл 2 задейства събитие 4.

При цикъл 3 имаме следното състояние:

Trakt3			
A	B	C	TNext
0	0	0	1

Събития

Поставя се индикатор END.

След като е поставен указател END, събитие 5 може да продължи в цикъл 4, където крайното състояние на модела се извежда. Няма по-нататъшни действия.

Проверка на събитията показва, че никакви по-нататъшни събития не се задействат при T = 0. Контролът на изпълнението автоматично ще увеличи времето на симулационния часовник към следващата точка от време, при която могат да се появят събития, в този случай при T = 1.

Събития

Също състоянието на модела в предишния момент време при цикъл 0 е валидно в момента T = 1.
Събитие 1 може да се обработи, базирайки се на това състояние в цикъл 0, след като TNext = 1 и неговото условие е изпълнено.
Чрез избиране обекта Protocol в каталога Protocols, изходът произведен от DISPLAY командите може да се види в прозореца на съдържанието. Те могат също да се изведат още с командата Print от менюто File.

Събития

Това дава следните резултати:

1 DISPLAY :	A = 0, B = 0, C = 0
2 DISPLAY :	A = 2, B = 0, C = 0
3 DISPLAY :	A = 2, B = 0, C = 0
4 DISPLAY :	A = 1, B = 2, C = 1
5 DISPLAY :	A = 0, B = 0, C = 0

Събития

Ред 1 произлиза от събитие 1 и показва оригиналното състояние при цикъл 0. Събитие 1 дава състоянието от цикъл 1. Двете събития 2 и 3 работят върху основното състояние при цикъл 1. По тази причина, редове 2 и 3, които произлизат от събития 2 и 3 са идентични.

Преходите между състоянията описани от събития 2 и 3 оказват ефект при цикъл 2. Събитие 4 така се основава на състоянието на модела, че се извежда в ред 4. Събитие 5 действа върху състоянието в цикъл 3 и генерира ред 5.

Събития

Най-важните факти могат да се обобщят:

- При всяка точка от време всички събития, които са позволени ще се изпълнят.
- Всички събития, които действат върху състояние на модела при определен цикъл, сами по себе си не променят състоянието на модела. Промените на състоянията, причинени от събития стават валидни при следващия цикъл.
- Тъй като за всички събития важи едно и също състояние на модела, редът по който те се обработват не е важен. Механизмът на цикъла в SIMPLEX MDL води до събития, независещи от реда, в който са описани.

Събития

Следния пример ще обясни как събитията не влияят на състоянието на модела по време на текущия цикъл, но оказват влияние на следващия цикъл:

Инициализираме променливата на състоянието A с A = 3.

```
STATE VARIABLE
DISCRETE
A (INTEGER) : = 3
```

След това имаме следното събитие:

```
ON ^ A = 3 ^
DO
        A ^ := 5;
        DISPLAY (“A = % d \ n”, A);
END
```

Събития

В цикъла 0 имаме A = 3 и събитието е стартирано. В този цикъл се изпълнява командата DISPLAY. Извеждането показва A = 3.

По време на същото събитие участва присвояването A = 5. То не оказва влияние до цикъл 1. Ако е нужно новата стойност да се изведе, тогава командата DISPLAY трябва да се изпълни един цикъл по-късно:

```
ON ^A = 3^
DO
    A ^ : = 5;
    SIGNAL Print;
END
ON Print
DO
    DISPLAY (“ A = % d \ n”, A);
END
```

При цикъл 0 са дадени стойности на променливата на състоянието A и индикатора Print. Новите стойности стават валидни. Сега DISPLAY показва изход A = 5.

Събития

Особено важно е, че указател за ON израз е валиден само за един цикъл. В моделът Test индикаторът End е поставен в събитие 4 и така става ефективен в следващия цикъл. Това води до стартиране на събитие 5. В края на цикъл 4, индикатора автоматично се нулира от стартиращия времеви контрол, така че събитие 5 няма да се изпълнява отново в следващия цикъл.

Събитието се изпълнява в цикъл, в който логическия израз е получил булевата стойност TRUE.

Събития

Едно събитие ще се изпълни в цикъл *n*, ако:

Предишен цикъл *n*-1: логическо условие FALSE

Цикъл *n*: логическо условие TRUE

Ако по време на последния цикъл стойността на логическото условие още веднъж се промени от FALSE на TRUE, тогава събитието ще се стартира още веднъж.

Събития

Обратно WHENEVER израз ще доведе до събитие, което се стартира във всеки цикъл, в който логическото условие има стойност TRUE.

С ON израз събитието ще се изпълни само в цикъл 1 и 5, тъй като само тогава стойността се променя от FALSE на TRUE.

Ако се използва WHENEVER израз, тогава събитието се стартира в цикли 1, 2 и 5. Следващата таблица демонстрира това:

Събития

Моделът QueueD

Ще бъде изграден модел на M/M/1 опашка, който се състои от източник, опашка, сървър и контейнер.

Събития

Източникът генерира идентични задачи през експоненциално разпределени интервали, със средна дължина 15 ВЕ. Така създадените задачи въвеждат опашката.

Сървърът съдържа 1 място, където работите могат да се обработват. Времето за обработка е експоненциално разпределено със средна стойност 10 ВЕ.

След обработката задачите напускат сървъра и отиват в контейнера. Когато задачите се съберат в контейнера, те се унищожават.

Събития

Моделът QueueD съдържа следните преходи между състояния:

ПРЕХОДИ	УСЛОВИЯ
Генериране на задача	Достигнато е време на пристигане
Задачата напуска опашката, влиза в сървъра и се обработва	Сървърът е свободен и опашката съдържа поне 1 задача
Задачата напуска сървъра и влиза в контейнера	Сървърът е зает и е достигнат края на времето за обработка
4 задачи са унищожени в контейнера	4 задачи са се натрупали в контейнера

Събития

Тъй като всички работи са идентични е достатъчно да се декларират променливите на състоянията, представящи броя на задачите в опашката, сървъра и контейнера.

Събития

BASIC COMPONENT QueueD

DECLARATION OF ELEMENTS

STATE VARIABLES

DISCRETE

```
NQueue   (INTEGER) := 0,
NServer  (INTEGER) := 0,
NSink    (INTEGER) := 0,
TArrive  (REAL)    := 0,
TWork    (REAL)    := 0,
Protocol (LOGICAL) := FALSE
```

Събития

```
RANDOM VARIABLES
   Arrive (REAL)     : EXPO
(Mean:=15,LowLimit:=0.5,UpLimit:=75),
   Work   (REAL)     : EXPO
(Mean:=10,LowLimit:=0.32,UpLimit:=50)

DYNAMIC BEHAVIOUR

# Създава се работа
WHENEVER T >= TArrive
DO
   NQueue^ := NQueue + 1;
   TArrive^ := T + Arrive;
   IF Protocol
   DO DISPLAY ("T= %f New customer \n",T); END
END
```

Събития

```
# Начало на обработката
WHENEVER (NServer = 0) AND (NQueue > 0)
DO
   NQueue^  := NQueue - 1;
   NServer^ := NServer + 1;
   TWork^ := T + Work;
   IF Protocol
   DO DISPLAY ("T= %f Customer enters
  server\n",T); END
END
```

Събития

```
# Край на обработката
WHENEVER (T >= TWork) AND (NServer = 1)
DO
   NServer^ := 0;
   NSink^   := NSink + 1;
   IF Protocol
   DO DISPLAY ("T= %f Customer leaves
  server\n",T); END
END
```

Събития

```
# Унищожаване на работа
WHENEVER NSink >= 4
DO
   NSink^ := 0;
   IF Protocol
   DO DISPLAY ("T= %f 4 customers
destroyed\n",T); END
END

END OF QueueD
```

Събития

	Време T=0	Време T=4	Време T=5
Цикъл 0	NQueue=0 Nserver=0 NSink=0 TArrive=0 TWork=0	NQueue=0 Nserver=1 NSink=0 TArrive=5 TWork=4	NQueue=0 Nserver=0 NSink=1 TArrive=5 TWork=4
Цикъл 1	NQueue=1 Nserver=0 NSink=0 TArrive=5 TWork=0	NQueue=0 Nserver=0 NSink=1 TArrive=5 TWork=4	NQueue=1 Nserver=0 NSink=1 TArrive=17 TWork=4
Цикъл 2	NQueue=0 Nserver=1 NSink=0 TArrive=5 TWork=4		NQueue=0 Nserver=1 NSink=1 TArrive=17 TWork=8

Събития

Състоянието на модела е определено от началните стойности в момент 0, цикъл 0. Това състояние на модела позволява събитие 1 да се стартира.

Като следствие от събитие 1 получаваме състоянието на модела при цикъл 1, който стартира събитие 2. Това събитие преобразува състоянието на модела от цикъл 1 в ново състояние в цикъл 2.

В момента T = 0 не могат да участват други условни събития. Контролът на изпълнение на Simplex3 търси зависими от времето събития. Събитието със следващото най-малко време е събитие 3, което е изпълнимо при T = 4. Симулационното време напредва към T = 4.

Събития

Състоянието в момента T = 4 и цикъл 0 е идентично на това при T = 0, цикъл 2. Събитието 3 променя това състояние в ново при цикъл 1.

От момента T = 4 не са възможни по-нататъшни условни събития, симулационното време отива към следващото събитийно време T = 5, когато събитие 1 може да се изпълни.

Събития

Първо за цикъл К =0 се изпълняват събитията, зависещи от времето. Новите стойности на дискретните променливи на състоянието формират новото състояние на модела в цикъла К=1. От това състояние са възможни няколко последващи събития, ако са изпълнени съответстващите условия. Симулационното време се увеличава автоматично от изпълнението на времевия контрол на Simplex3.

TArrive^ := T + Arrive;

TWork^ := T + Work;

Събития

Крайни, дискретно-времеви автомати.

Крайните автомати (КА) се характеризират от крайна редица от стойности на променливите на състоянията и зависими променливи, и от преходите между състоянията, които се извършват само в дискретни моменти от време.

Като пример за КА ще разгледаме автомат за продажба на цветя.

Обхватът за входния набор *X* се състои от 4 елемента:

X = {x0, x1, x2, x3}

{nothing, fCash, rCash, fill}

Събития

Обхватът за променливата на състоянието Z е набор от 4 елемента:

Z = {z0, z1 ,z2, z3}

{0 bunches, 1 bunch, 2 bunches, 3 bunches}

Обхватът на външната променлива Y има следните стойности:

Y = {y0, y1, y2}

{empty, cash, flower}

Локалната функция на преходите на КА може да се опише чрез използване на матрица на преходите.

Събития

Локална функция на преходите – f

	Z_0	Z_1	Z_2	Z_3	Функция на автомата
X_0	Z_0	Z_1	Z_2	Z_3	Няма пари – старо състояние
X_1	Z_0	Z_1	Z_2	Z_3	Неточни пари – старо състояние
X_2	Z_0	Z_0	Z_1	Z_2	Точни пари – намаление на броя букети с 1, ако е възможно

Събития

Локалната функция на преходите определя новото състояние в цикъла k_{i+1} в момента от време t_n чрез старото състояние. Имаме:

$z(t_n, k_{i+1}) = f\,(z(t_n, k_i), x(t_n,k_i))$

Това означава, че машината, която е в състояние z_2 при цикъл k_i въвежда състояние z_1, когато входът е равен на x_2. Новото състояние z_1 е въведено в цикъл $к_{i+1}$.

z2: Съдържание на машината 2 букета

x2: Вход коректни пари

z1: Съдържание на машината 1 букет

Събития

Изходната функция g описва изхода на машината в състояние z с вход x в цикъла ki.
Може да се види, че входа и изхода са поставени в един и същи цикъл. Преходът на състоянията за z се случва един цикъл по-късно. Имаме:

Y(tn,ki) = g(z(tn,ki),x(tn,ki))

Събития

Изходна функция g:

	Z_0	Z_1	Z_2	Z_3	Поведение на автомата
X_0	Y_0	Y_0	Y_0	Y_0	Няма пари – няма изход
X_1	Y_1	Y_1	Y_1	Y_1	Неточни пари – връщане на пари
X_2	Y_1	Y_2	Y_2	Y_2	Точни пари – 1 букет, ако е възможно
X_3	Y_0	Y_0	Y_0	Y_0	Ново зареждане – няма изход

Събития

При цикъл k0 е налице началното състояние. В цикъла k1 са известни входът X и изходът Y, които са базирани на старото състояние.

След обработване на входа и изпълнение на изхода в следващия цикъл 2 се определя новата стойност на променливата на състоянието.

Събития

```
BASIC COMPONENT Vending

 LOCAL DEFINITIONS
   VALUE SET  Input: ('nothing', 'f_Cash', 'r_Cash', 'fill')
   VALUE SET Output: ('empty', 'Cash', 'Flowers')
 DECLARATION OF ELEMENTS
   STATE VARIABLES
     X     (Input)   := 'nothing',  # Вход
     Z     (INTEGER) := 3,          # Състояние
     TStep (INTEGER) := 0           # Напредване на времето
```

Събития

```
DEPENDENT VARIABLES
   Y (Output) := 'empty'     # Изход
  RANDOM VARIABLES
   Rand (INTEGER) : IUNIFORM (LowLimit := 0, UpLimit :=4)
  TRANSITION INDICATORS
   StateCh,                  # Промяна на състояние
   Print                     # Отпечатване на ново състояние
```

Събития

```
DYNAMIC BEHAVIOUR
 WHENEVER T >= TStep
 DO     DISPLAY("TStep %d \n", TStep);
    DISPLAY("Old state: Number of bunches = %d \n", Z);
 IF Rand = 0          # случаен вход
      DO X^ := 'nothing';  END
    ELSIF Rand = 1
      DO X^ := 'f_Cash';  END
    ELSIF Rand = 2
      DO X^ := 'r_Cash';  END
    ELSIF Rand = 3
      DO X^ := 'r_Cash';  END
    ELSE
      DO X^ := 'fill'; END
      SIGNAL StateCh;
      TStep^ := TStep + 1;
 END
```

Събития

```
# Промяна на състоянието
ON StateCh
DO
   DISPLAY ("Input X = %d  Output Y = %d \n", X, Y);
   IF X = 'fill'
   DO
     Z^ := 3;
   END
   ELSIF X = 'r_Cash'
   DO
     IF Z > 0
     DO Z^ := Z - 1;  END
   END
   SIGNAL Print;
END
```

Събития

```
# Алгебрични уравнения за изход
IF X = 'fill'
   DO Y := 'empty';  END
ELSIF X = 'f_Cash'
   DO Y := 'Cash';  END
ELSIF X = 'r_Cash'
   DO
     IF Z > 0
       DO Y := 'Flowers';  END
     ELSE
       DO Y := 'Cash';  END    END
```

Събития

```
ELSE
   DO Y := 'empty';  END

 # Отпечатване на новото състояние
 ON Print
 DO
   DISPLAY ("New State: Number of bunches Z =
     %d \n\n", Z);
 END

 END OF Vending
```

Събития

Началните стойности при T = 0 и цикъл 0 са :

X = ‘nothing’, z = 3, y = ‘empty’

Състоянието на модела се определя изцяло от променливите на състоянията x и z. Новото състояние при всяка точка от време се определя от предишното състояние чрез локалната функция на преходите f.

Във всеки момент от време съставните променливи при цикъл 0 са непроменени. В следващия цикъл 1 променливата X има съответна стойност.

Събития

Променливата на състоянието z получава новата си стойност в цикъл 2.

Изходът у произтича от състоянията на променливите x и z. Затова у е зависима променлива. Всеки път, когато x и z променят стойностите си контролът на изпълнението в Simplex3 осигурява, че зависимата променлива в изходната функция g се преизчислява.

Събития

Може да се види, че във всеки момент от време взима участие пълна процедура, състояща се от вход, изход и промяна на състояние. След като всички промени на условията на състоянието се изпълняват, времето се придвижва напред и започва нова процедура.

Изходите, създадени от DISPLAY командата могат да се проверят чрез избиране на обект Protocol в каталога Protocols в прозореца на съдържанията. Може да се изведе чрез командата Print от менюто File.

Забележка: В протокола се записват стойностите на VALUE SET променливите под формата на техните вътрешни числени представяния, дефинирани от потребителя, т.е. y = 2 вместо y = ‘flowers’.

Събития

Следващият пример показва първите 5 времеви стъпки на симулационното изпълнение:

>> T=0.000 : Start of simulation

DISPLAY: Tstep 0

DISPLAY: Old state: Number of bunches = 3

DISPLAY: Input X= 2 Output Y =2

DISPLAY: New State: Number of bunches Z = 2

Събития

DISPLAY: Tstep 1
DISPLAY: Old state: Number of bunches = 2
DISPLAY: Input X = 2 Output Y =2
DISPLAY: New State: Number of bunches Z =1

DISPLAY: Tstep 2
DISPLAY: Old state: Number of bunches = 1
DISPLAY: Input X = 1 Output Y =1
DISPLAY: New State: Number of bunches Z =1

Събития

DISPLAY: Tstep 3

DISPLAY: Old state: Number of bunches = 1

DISPLAY: Input X = 3 Output Y =0

DISPLAY: New State: Number of bunches Z =3

DISPLAY: Tstep 4

DISPLAY: Old state: Number of bunches = 3

DISPLAY: Input X = 2 Output Y =2

DISPLAY: New State: Number of bunches Z =2

Събития

DISPLAY: Tstep 5

DISPLAY: Old state: Number of bunches = 2

DISPLAY: Input X = 2 Output Y =2

DISPLAY: New State: Number of bunches Z =1

>> T = 5.000 : End time reached

Забележка:

Циклите гарантират независимостта на реда на събитията. Това означава, че събитията могат да се напишат в произволен ред и да се разпределят произволно между компонентите.

Събития

Разделяне на пространството на състоянията.

Налице са следните възможности за описание на синтаксиса за динамично поведение

statement ::= algebric_equation
| differential_equations
| region_defining_statement
| event_ defining_statement

Като допълнение на алгебричните уравнения, диференциалните уравнения и събития има езикови конструкции, разделящи пространството на състоянията. Разделянето на пространството на състоянията означава определяне на различни динамики, зависещи от състоянието на модела.

Събития

Ключовите думи IF, ELSEIF и ELSE се използват, за да специфицират секциите с различни динамики.

Да разгледаме модела Plane, който описва движенията на топка върху две равнини под ъгъл една с друга.

Състоянието на модела в този случай е описано от двете съставни променливи *x* и *v*. Пространството на състоянията е двумерно.

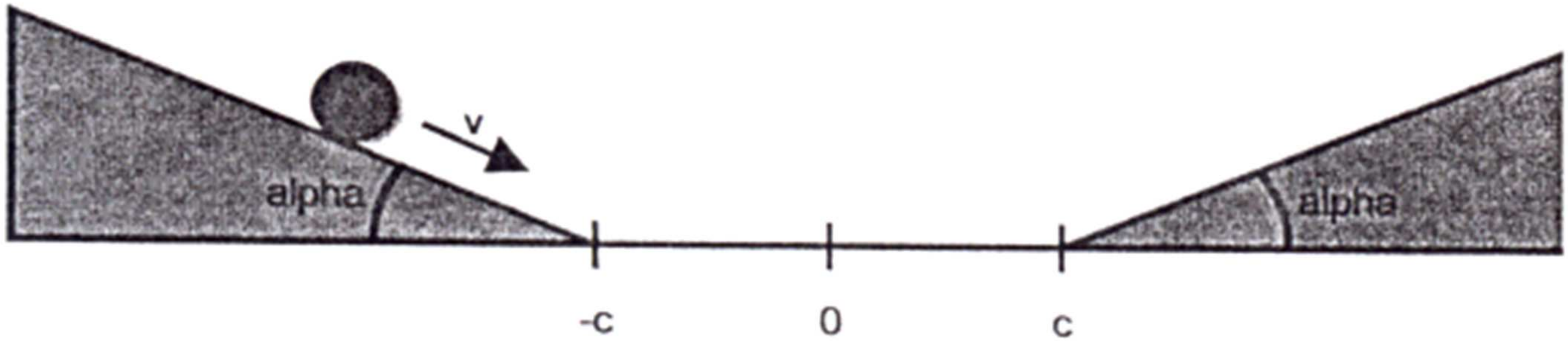
Трите условия, разделящи пространството на състоянията на 3 части:

$x < -c$

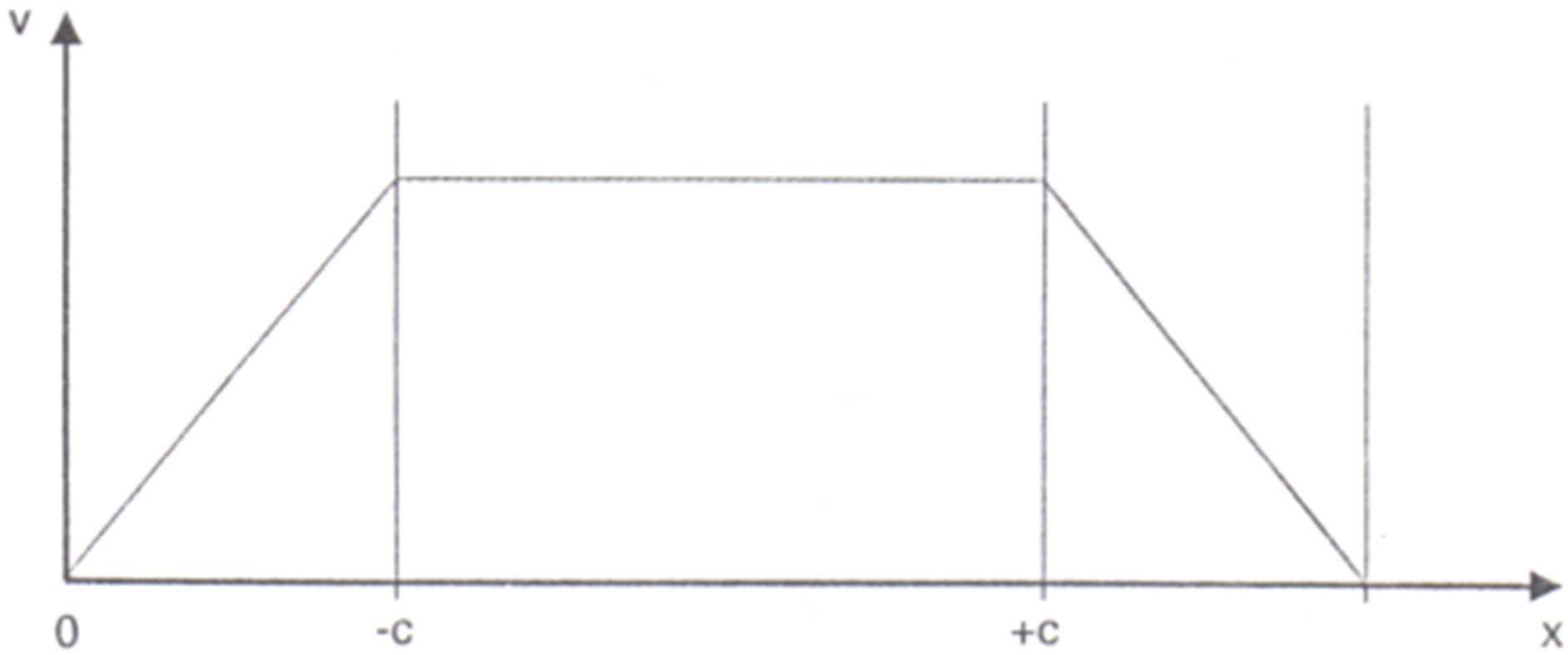
$x > c$

$-c <= x <= c$

Събития



Събития



Събития

```
BASIC COMPONENT Plane

 USE OF UNITS
  TIMEUNIT = [s]

 DECLARATION OF ELEMENTS
   CONSTANTS
     g (REAL[m/s^2]) := 9.81 [m/s^2], # гравитационно ускорение
     alpha (REAL)    := 0.52,         # ъгъл на наклон
     c (REAL[m])     := 1.5 [m]       # хоризонтална секция

   STATE VARIABLES
  CONTINUOUS
    x (REAL[m])   := -5 [m],        # x-координата
    v (REAL[m/s]) := 0 [m/s]        # скорост
```

Събития

DYNAMIC BEHAVIOUR

```
# ляво-наклонена равнина
IF x < -c
DO
  DIFFERENTIAL EQUATIONS
    v' := g * SIN(alpha);
    x' := v * COS(alpha);
  END
END
```

Събития

```
# дясно-наклонена развнина
ELSIF x > c
DO
  DIFFERENTIAL EQUATIONS
    v' := -g * SIN(alpha);
    x' :=  v * COS(alpha);
   END
END

# хоризонтална равнина
ELSE
DO
  DIFFERENTIAL EQUATIONS
    v' := 0 [m/s^2];
    x' := v;
  END
END

END OF Plane
```

Събития

Синтаксиса е:

dynamic_behaviour : : = DYNAMIC BEHAVIOUR
    statement_sequence
statement_sequence : : = statement { statement }
statement : : = algebric_equation
    | differential_equations
    | region_defining_statement
    | event_ defining_statement
region_defining_statement : : = IF expression
    DO statement_sequence END
    {ELSEIF expression
    DO statement_sequence END}
    [ELSE expression
    DO statement_sequence END]

Събития

Region_defining_statement може да съдържа алгебрични уравнения, диференциални уравнения, събития и разделяния на пространството на състоянията.

Моделът Vending е пример за разделяне пространството на състоянията с алгебрични уравнения. В зависимост от променливата на състоянието x се определя зависимата променлива y:

```
IF X = ‘fill’
      DO Y :=  ‘empty’; END
ELSEIF X = ‘f_Cash’
      DO    Y : = ‘Cash’; END
ELSEIF X = ‘r_Cash’
      DO
           IF Z > 0
                  DO Y : = ‘Flowers’; END
          ELSE
                  DO Y : = ‘Cash’; END
    END
ELSE
   DO Y : = ‘empty’; END
```

Събития

Масиви.

Всички количества могат да се дефинират като масиви до три измерения. Всяка стойност е достъпна чрез един или повече индекси. Индексите започват от 1 и стигат до стойността, определена в декларацията. Най-големият индекс е броя на елементите на масива в съответното измерение.

Пример:

ARRAY [3] [5] X (Real) # двумерен масив с 15 елемента

Събития

Индекси.

Всеки индивидуален елемент се идентифицира чрез индекс или списък от индекси.

Синтаксис:

Index_identifier : : = [index_or_index_set
{ index_or_index_set }]

index_or_index_set : : = ‘[‘expression’]’
| ‘{‘index_set’}’

index_set : : = basis_set [‘|’ expression]

basis_set : : = index_range
| identifier OF index_range
| ALL
| ALL identifier

Index_range : : = unsigned_integer
unsigned_integer ‘..’ unsigned_integer

unsigned_integer : : = unsigned_number
| identifier

Събития

Индивидуален индексен израз се затваря в квадратни скоби, а набор от индекси във фигурни скоби. Индексен израз е числов израз от тип INTEGER.

Набор от индекси отбелязва непрекъсната област от цели числа. Тази област може да се ограничи от условие, което следва след символа ‘|’.

Непрекъсната област от индексен набор може да се даде чрез долна и горна граници или с ключова дума ALL, която отбелязва всички възможни индекси на идентификатора в съответното измерение.

Идентификаторът може да се използва заедно с индексния набор за представяне на всички индивидуални индекси.

Събития

Моделни величини са:

- Константи;
- Случайни променливи;
- Зависими променливи;
- Сензорни променливи.

Всички моделни величини могат да се представят чрез масиви.

Събития

Примери:

Елемент на масив се идентифицира чрез индекса си

```
DECLARATION OF ELEMENTS
    STATE VARIABLES
    DISCRETE
                ARRAY [3] X (INTEGER) : = 0,
                ARRAY [3] [5] Y (INTEGER) : = 1
DYNAMIC BEHAVIOUR
     WHENEVER X[1] > = 0
      DO
             X[1]^ : = 2;
             Y[1] [1]^ : = 2;
END
```

Събития

Вместо индекс може да се използва израз, за да се оцени индекса

```
DECLARATION OF ELEMENTS
    CONSTANTS
     k (INTEGER) : = 1
    STATE VARIABLES
    DISCRETE
            ARRAY [3] X (INTEGER) : = 0
                  DEPENDENT VARIABLES
  DISCRETE
                ARRAY [3] [5] Y (INTEGER)
DYNAMIC BEHAVIOUR
   Y [k] [k+1] : = X[2*k+1];
WHENEVER X[1] >= 0
DO
  X[k]^ := 5;
END
```

Израза за оценяване на индекс трябва да даде INTEGER стойност. Излизане извън границите на масива води до грешка при изпълнението.

Събития

Вместо отделен индекс може да се използва набор от индекси. Промяната на състоянието се извършва за всички индекси:

```
DECLARATION OF ELEMENTS
    STATE VARIABLES
    DISCRETE
      ARRAY [6]     X(INTEGER) : = 0,
      ARRAY [3]   [5]  Y(INTEGER) : = 0
DYNAMIC BEHAVIOUR
       WHENEVER X[1] >= 0
     DO
          X {1..3}^  : = 1;
          X{4..6}^ : = 2;
         Y{1..2} {3..5}^ : = 3;
END
```

Събития

Ключовата дума ALL отбелязва всички възможни индекси

```
DECLARATION OF ELEMENTS
STATE VARIABLES
     DISCRETE
          ARRAY[5] X (INTEGER) : = 0;
ARRAY [3] [5]   Y   (INTEGER) : = 1
DYNAMIC BEHAVIOUR
WHENEVER X[1] >= 0
      DO
                X {ALL}^ : = Y {ALL} [1];
END
```

В това събитие се извършват 3 присвоявания, като се препокриват три елемента от вектора X.

Събития

Представя се идентификатор на индексите. Този индекс може да се използва при заявлението:

```
DECLARATION OF ELEMENTS
STATE VARIABLES
CONTINUOUS
        ARRAY [5] X (REAL) : = 0,
        ARRAY [3] [5]   Y   (REAL) : = 1
DYNAMIC BEHAVIOUR
  DIFFERENTIAL EQUATIONS
  X {i OF 1..2}' : = 3 * i;
  Y {ALL} {Index OF 1..3}' : = X [Index];
  Y {ALL} {Index OF 4..5}' : = X [Index] + 1;
END
```

Някои елементи на масивите X и Y са дефинирани като диференциални уравнения. Идентификаторите i и присъстват в дясната страна на диференциалното уравнение.

Събития

Пълният набор от индекси може да се отбележи също чрез ключовата дума ALL

```
DECLARATION OF ELEMENTS
STATE VARIABLES
CONTINUOUS
            ARRAY [5] X (REAL) : = 0,
            ARRAY [5] [3]   Y   (REAL) : = 1
DYNAMIC BEHAVIOUR
  DIFFERENTIAL EQUATIONS
      X {ALL j}’: = j *X {j};
      Y {k} {ALL k}’: = k * X[k];
END
```

Диференциалните уравнения се дефинират за всички елементи от масива X. В масива Y са засегнати елементите Y[k] [1], Y[k] [2] и Y[k] [3].

Събития

Възможно е да се ограничи индексния набор. Засягат се само онези индекси, за които логическият израз има стойност TRUE.

```
DECLARATION OF ELEMENTS
      STATE VARIABLES
      DISCRETE
                  ARRAY [5] X       (INTEGER) : = 0,
                  ARRAY [5] [3] Y (INTEGER) : = 1
DYNAMIC BEHAVIOUR
 WHENEVER X [1] = 0
  DO
                   X {ALL i | Y [i] {1} > 0}^ : = 1;
END
```

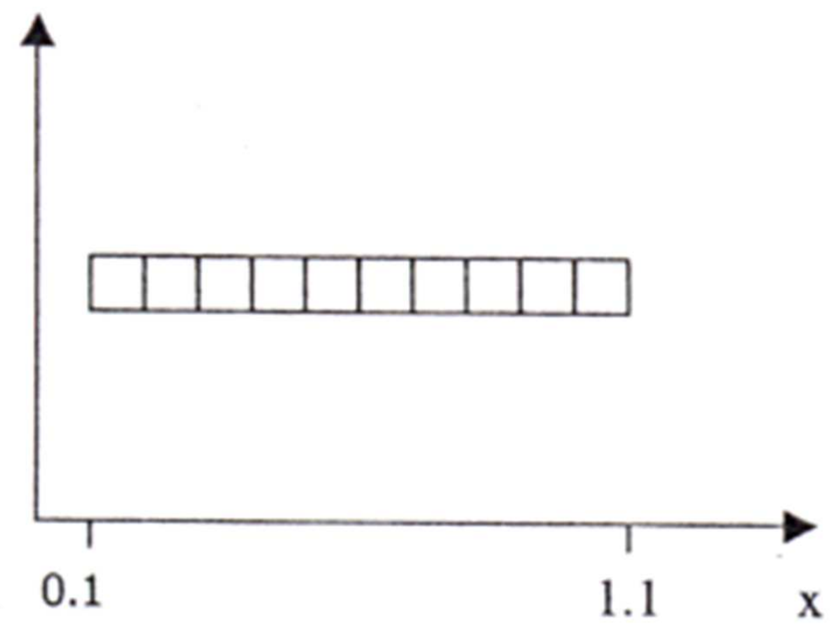
Събития

Параболични частни диференциални уравнения

Избираме уравнението на топлопроводимостта за хомогенна пръчка като пример за параболично диференциално уравнение. Прието е, че страните на пръчката са изолирани. Промяна на топлинната енергия се регистрира само в двата края на пръчката.

Събития

Фигурата показва тънка пръчка с дължина 1, чиито краища са разположени върху оста X на 0.1 и 1.1. Страните на пръчката са изолирани, така че не се обменя никаква енергия с околната среда. Това означава, че топлина може да се движи само в пространството X. Затова това е едномерен проблем.



Събития

Левият край на пръчката при x = 0.1 се пази на константната температура от 0^0 C. В началото – цялата пръчка има температура u = 0. В момента T= 0 температурата на десния край при x = 1.1 е u = 2.

Очаква се, че пръчката се затопля бавно и температурата накрая ще достигне устойчиво състояние, в което левият край има температура u = 0^0 и десния край има температура u = 20 . В момента T = 10 температурата все още е далеч от равновесие. В момента T = 500 е налице гладко линейно разпределение.

Събития

Температурата може да се наблюдава в една избрана точка от пръчката по време на симулацията.

Събития

Топлинният обмен е описан от следните частни диференциални уравнения:

$$\frac{\partial u}{\partial t} = D \cdot \frac{\partial^2 u}{\partial x^2}$$

Това уравнение е частен случай от по-общото уравнение.

$$\frac{\partial u}{\partial t} = g\left(t, u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right)$$

Събития

Промяната по всяко време *t* зависи от диференциалните коефициенти

$$\frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}$$

Двата диференциални коефициента могат да се приближат чрез разлики.

За да направим това разделяме *X* направлението на *n* интервала с дължина *k*. Триточкова апроксимация се използва за всяка точка от мрежата.

Събития

Диференциален коефициент от първи ред:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2k} \qquad i \neq 1, n$$

За лявата (дясната) граница имаме:

$$\left(\frac{\partial u}{\partial x}\right)_1 = \frac{-3u_1 + 4u_2 - u_3}{2k}$$

$$\left(\frac{\partial u}{\partial x}\right)_n = \frac{u_{n-2} - 4u_{n-1} + 3u_n}{2k}$$

Събития

Диференциален коефициент от втори ред:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{k^2} \qquad i \neq 1, n$$

Събития

Където с u_i е отбелязана стойността на функцията в момента време t_j за x-координатата x_i:

$u_i = f(x_i, t_j)$

Приемаме следните начални условия за топлинния трансфер:

$u(x, t) = 0 \quad 0.1 <= x < 1.1 \quad t = 0$

$u(x, t) = 2 \quad x = 1.1 \quad t = 0$

В момента време $t = 0$ температурата u е равна на нула в цялата пръчка. Само на десния край при $x = 1.1$ имаме $u = 2$.

Събития

Приемаме за граничните условия, че температурата в краищата на пръчката е константна през време:

$u(x, t) = 0 \quad x = 0.1 \; t >= 0$

$u(x, t) = 2 \quad x = 1.1 \; t >= 0$

Ще изучим промените на температурата в пръчката при дадените начални и гранични условия.

Разделяме пръчката на 10 интервала с дължина 0.1. Получаваме 11 координатни точки.

За всяка точка имаме обикновено диференциално уравнение от първи ред:

Събития

$$\frac{du_i}{dt} = D.\left(\frac{\partial^2 u}{\partial x^2}\right)_i$$

Събития

За което се изпълнява: D = 0.001 {m^2/min]

Диференциалния коефициент $\dfrac{\partial^2 u}{\partial x^2}$ ще бъде

приближен чрез метод описан по-нататък.

Желаем да изчислим промяната на температурата на 11 различни региона чрез диференциално уравнение от първи ред.

Събития

Имаме следната процедура:

- Изчисляваме диференциалния коефициент:

$$\frac{\partial^2 u}{\partial x^2}$$

базиран на състоянието на модела в момента t.

Събития

Първо се оценява диференциалният коефициент

$$\frac{\partial^2 u}{\partial x^2}$$

за всяка точка i от мрежата чрез приближение с три точки. Това е записано във вектора u2 по следния начин:

Събития

$$u\_2[1] = \frac{\partial^2 u(x,t)}{\partial x^2}, \quad x = 0.1$$

$$u\_2[2] = \frac{\partial^2 u(x,t)}{\partial x^2}, \quad x = 0.2$$

$$u\_2[11] = \frac{\partial^2 u(x,t)}{\partial x^2}, \quad x = 1.1$$

Събития

С така определените диференциални коефициенти

$$\frac{\partial^2 u}{\partial x^2}$$

може да се опише диференциално уравнение.

От описанието на модела се вижда, че всички точки от мрежата са инициализирани с нула.

Първата промяна на състояние е чрез събитие, в което са поставени началните стойности

$u(x,t) = 2$ за $x = 1.1$ $t = 0$.

Събития

- Единадесетте диференциални коефициента

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i$$

i = 1. .11

са дефинирани от стойността на променливата на състоянието u. Те ще се изчислят чрез алгебрични уравнения. Отделно се третират граничните условия при x = 0.1 и x=1.1.

Събития

Втората частна производна от интервала u_2[i] може да се използва, за да определи диференциалния коефициент u[i]'.

Забележка:

Стабилността на численото интегриране на частните диференциални уравнения изисква повече грижи. За уравнението на топлопроводимостта стабилността зависи от следната стойност:

$$\alpha = \frac{D.\Delta t}{\Delta x^2}$$

където трябва да е изпълнено $\alpha < 0.1$

События

```
BASIC COMPONENT Rod

 USE OF UNITS
   TIMEUNIT = [min]

 DECLARATION OF ELEMENTS

   CONSTANTS
     D (REAL[m^2/min]) := 1.0 E-3 [m^2/min]
```

Събития

STATE VARIABLES

CONTINUOUS

ARRAY[11] u (REAL[Cel]) := 0 [Cel]

температура в точка x

DEPENDENT VARIABLES

CONTINUOUS

ARRAY[11] u_2 (REAL[Cel/m^2]) := 0 [Cel/m^2]

2ра производна в точка x

Събития

DYNAMIC BEHAVIOUR

```
 u_2  [1]          :=
(u[3]   - 2*u [2] + u [1] ) / (POW(0.1, 2) * 1[m^2]);
 u_2 [11]          :=
(u[11]  - 2*u[10] + u [9] ) / (POW(0.1, 2) * 1[m^2]);
 u_2 {i OF 2..10} :=
(u[i+1] - 2*u [i] + u[i-1]) / (POW(0.1, 2) * 1[m^2]);
```

Събития

```
ON START
DO
   u[11]^ := 2 [Cel];
END
DIFFERENTIAL EQUATIONS
  u {i OF 1..10}' := D * u_2[i];
  u  [1]'         := 0 [Cel/min];
  u [11]'         := 0 [Cel/min];
END
END OF Rod
```

Събития

Могат да се използват стойностите по подразбиране на контролните параметри, за да се изпълни симулацията на модела Rod.

В примера симулацията се изпълнява до T = 500. В прозореца Analyse and present функцията TimeCut се избира заедно с подходяща точка от време. Динамичните редове Obs#/Rod/u[10] и Obs1#/Rod/u[11] се поставят на края на списъка.