Conversión de modelos PowerDEVS al lenguaje Modelica

Tesinista: Luciano Andrade
Director: Federico Bergero, Co-Director: Ernesto Kofman

2 de febrero de 2015

Índice general

Resumen

En este trabajo se describe la implementación de una aplicación para convertir modelos descriptos en la herramienta PowerDEVS a modelos en el lenguaje Modelica, más especificamente en μModelica, con el fin de aprovechar la velocidad de simulación del 'QSS Solver', permitiendo describir las simulaciones en el entorno PowerDEVS y ejecutando las simulaciones en 'QSS Solver'

Capítulo 1

Introducción

1.1. Motivación y Objetivos

PowerDEVS es una herramienta de simulación de sistemas híbridos, basado en el formalismo DEVS, con una interfaz grafica orientada a bloques, donde los bloques pueden ser conectados entre si, modificado los parametros, posibilidad de conectarse con el entorno Scilab para poder utilizar expresiones y herramientas de cálculo provistas por este entorno.

La resolución de ecuaciones diferenciales ordinarias, requiere el uso de métodos de integración numérica. Todos los algoritmos tradicionales de integración se basan en la discretización de la variable independiente (que generalmente representa el tiempo). Las rutinas que implementan estos algoritmos, se denominan solvers y existen gran variedad de implementaciones de los mismos en diferentes lenguajes de programación. Los Métodos de Integración Numérica QSS (Quantized State System), a diferencia de los métodos de integración tradicionales, realizan la discretización sobre las variables de estado. En consecuencia, convierten los sistemas continuos en sistemas de eventos discretos, y tienen grandes ventajas para simular sistemas con discontinuidades. Si bien PowerDEVS, implementa la totalidad de los algoritmos de QSS, resultan ineficientes, dado que malgastan gran parte de la carga computacional en la transmisión de eventos entre submodelos.

Para solventar este hecho se desarrollo una familia de QSS stand-solver, el cual requiere un modelo descripto en lenguaje C el cual contiene las equaciones difereciales, las funciones de cruce de cero asi como la información estructural requerida por los algoritmos QSS. Estos solvers obtienen una mejora de performace de hasta un orden de maginitud comparado con otras implementaciones DEVS. Sobre este se desarrollo una herramineta la cual genera a partir de un modelo μ-Modedelica (un subconjunto del lenguaje Modelica) el modelo requerido para el QSS solver.

Con el objetivo de utilizar los mejoras de velocidad y mantener un entorno amigable con el usuario, se creo una herramienta capas de convertir

un modelo PowerDEVS en un modelo μ-Modelica.

1.2. Trabajo relacionado

¿Cuales?

1.3. Alcance

DEVS, Discrete Event System Specification (Especificación de Sistemas de Eventos Discretos), es un formalismo modular y jerarquico para modelar y analizar sistemas que pueden ser de eventos de tiempo discreto mediante tablas de transición, y con estados continuos mediantes que pueden ser descriptos mediantes equaciones diferenciales.

En el formalismo clasico DEVS, los modelos atómicos capturan el comportamiento del sistema, mientras los modelos Acoplados describen la estructura del sistema.

En particular los modelos atómicos en PowerDEVS son descriptos en clases C++, mientras que la estructura se encuentra definida en archivos pds y pdm.

Modelica es un lenguaje de modelado, orientado a objetos, declarativo, para el modelado orientado a componentes de sistemas complejos.

Para poder realizar nuestro objetivo es necesario primero contar con un modelo en modelica para cada atómico PowerDEVS que deseemos convertir.

De esto se desprenden las siguientes limitaciones importantes:

- La semántica de los modelos convertidos depende de los modelos equivalentes a los DEVS atómicos
- Solamente podemos convertir modelos cuyos componentes atómicos estén o puedan ser convertidos a μmodelica.

Capítulo 2

Conceptos Previos

En este capítulo introducimos algunos conceptos básicos, necesarios para poder comprender este trabajo.

2.1. Modelado y Simulación

Modelado y Simulación de un Sistema es el proceso por el cual se desarrolla un modelo, el cual es luego simulado, de forma de obtener datos sobre el sistema. El modelo debe conservar las principales características del sistema, pero al mismo tiempo ser significativamente más simple, de forma que al momento de simularlo sea más eficaz utilizar la simulación que el sistema en si.

2.1.1. Sistemas Continuos y Discretos

Se considera un sistema continuo si las variables de este son conocidas en cada instante de tiempo, mientras que se considera discreto si las variables son conocidas en instantes de tiempo determinados.

En general los sistemas en estudio serán continuos, pero deberemos utilizar sistemas discretos puesto que la simulación en computadora así lo requiere, puesto que la misma computadora es un sistema discreto.

2.1.2. Métodos de Integración numérica

Un sistema continuo puede ser descripto por un modelo en espacios de estados de la forma:

$$x(t) = f(x(t), u(t)) \tag{2.1}$$

donde $x \in \Re^n$ es el vector de estados, $u \in \Re^m$ es una función de entradas conocidas, t representa el tiempo y con sus condiciones iniciales:

$$x(t = t0) = x0 \tag{2.2}$$

Sea $x_i(t)$ la trayectoria del estado i-esimo expresada como función de tiempo simulado. Mientras que la ecuación (2.1) no contenga discontinuidades $x_i(t)$ será una función continua con derivada continua. Esta puede ser aproximada con la precisión deseada mediante series de Taylor en cualquier punto de su trayectoria

Denominando t^* al instante de tiempo en torno al cual se aproxima la trayectoria mediante una serie de Taylor, y siendo $t^* + h$ el instante de tiempo en el cual se quiere evaluar la aproximación, entonces, la trayectoria en dicho punto puede expresarse como sigue:

$$x_i(t^* + h) = x_i(t^*) + \frac{dx_i(t^*)}{dt} \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \cdots \tag{2.3}$$

Reemplazando con la ecuación de estado (2.1), la serie (2.3) queda:

$$x_i(t^* + h) = x_i(t^*) + f_i(t^*) \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \cdots \tag{2.4}$$

Los distintos algoritmos de integración difieren en la manera de aproximar las derivadas superiores de f y en el número de términos de la serie de Taylor que consideran para la aproximación.

2.2. Formalismo DEVS

DEVS es un formalismo para modelar y analizar sistemas de eventos discretos (es decir, sistemas en los cuales en un lapso finito de tiempo, ocurren una cantidad finita de eventos). Un modelo DEVS puede ser visto como un autómata que procesa una serie de eventos de entrada y genera una serie de eventos de salida. Este procesamiento está regido por la estructura interna de cada una de las partes que componen el modelo general. Un modelo DEVS está descripto por dos clases de componentes, modelos atómicos y modelos acoplados.

2.2.1. Atómicos

Un modelo atómico representa la unidad ïndivisible"de especificación, en el sentido que es la pieza fundamental y más básica de un modelo DEVS. Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{2.5}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico, es decir un evento de entrada tiene como valor un elemento del conjunto X.

- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo, en todo momento el atómico está en un estado dado, que es un elemento del conjunto S.
- ta es una función $S \rightarrow R^+$, que indica cuánto tiempo el modelo atómico permanecerá en un estado dado, si es que no se recibe ningún evento de entrada. Esta función puede asociarse también al tiempo de vida de un estado.
- δ_{int} es una función $S \rightarrow S$, que indica la dinámica del sistema en el momento que el modelo atómico realiza una transición interna. Sería el análogo a una tabla de transición en otros autómatas.
- δ_{ext} es una función $(S \times R^+ \times X) \rightarrow S$, que indica el cambio de estado ante la presencia de un evento externo.
- λ es una función $S \rightarrow Y$ que indica qué evento se debe emitir al salir de un estado dado.

2.2.2. Acoplados

2.2.3. Modelos Vectoriales

BK11

2.3. Métodos de integración QSS

QSS paper

2.4. PowerDEVS

Powerdevs Paper

2.5. Modelica

2.6. QSS Stand Alone Solver

QSSPaper

2.6.1. μModelica

Capítulo 3

Conversión de modelos DEVS

3.1. Modelos Atómicos

Cómo se traducen (es conocimiento del modelador, no automático)

3.2. Modelos Vectoriales

Consideraciones y anotaciones en modelica, modificaciones

3.3. Modelos Acoplados Planos

Modelos acoplados solo con modelos atómicos adentro.

Mencionar el algoritmo (traducción de conexiones y "aplanado" de cada uno de los atómicos hijos")

3.4. Equivalencia semántica de la conversión

3.5. Modelos Acoplados Jerárquicos

Explicar cuándo se utilzan y que resolvemos el problema aplanando los acoplados

3.5.1. Algoritmo de aplanado

Describir el algoritmo para PDS

3.6. Comparación de performance

¿No deberia estar despues de .Ejemplos de Aplicación¿

Capítulo 4

Detalles de Implementación

API Powerdevs, AST Modelica Traverser Modelica Transformer

4.1. Ejemplos de Aplicación

tamaños de las vectores y comparativas de

4.1.1. Vector/airs

4.1.2. Vector/lcline

4.2. Conclusiones y Trabajo a futuro

Bibliografía

[1] Federico Bergero, Xenofon Floros, Joaquín Fernández, Ernesto Kofman, and François E. Cellier. Simulating Modelica models with a Stand–Alone Quantized State Systems Solver. In *9th International Modelica Conference*, 2012. Aceptado.

[2] Federico Bergero and Ernesto Kofman. PowerDEVS: a tool for hybrid system modeling and real-time simulation. *Simulation*, 87(1–2):113–132, 2011.

[3] François Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer, New York, 2006.

[4] Joaquín Fernández and Ernesto Kofman. Implementación autónoma de métodos de integración numérica QSS. Technical report, FCEIA - UNR, Rosario, Argentina, 2012.

[5] Xenofon Floros, Federico Bergero, Francois E. Cellier, and Ernesto Kofman. Automated Simulation of Modelica Models with QSS Methods - The Discontinuous Case. In *8th International Modelica Conference*, March 2011.

[6] Peter Fritzson and Vadim Engelson. Modelica - A Unified Object-Oriented Language for System Modelling and Simulation. In *ECOOP*, pages 67–90, 1998.