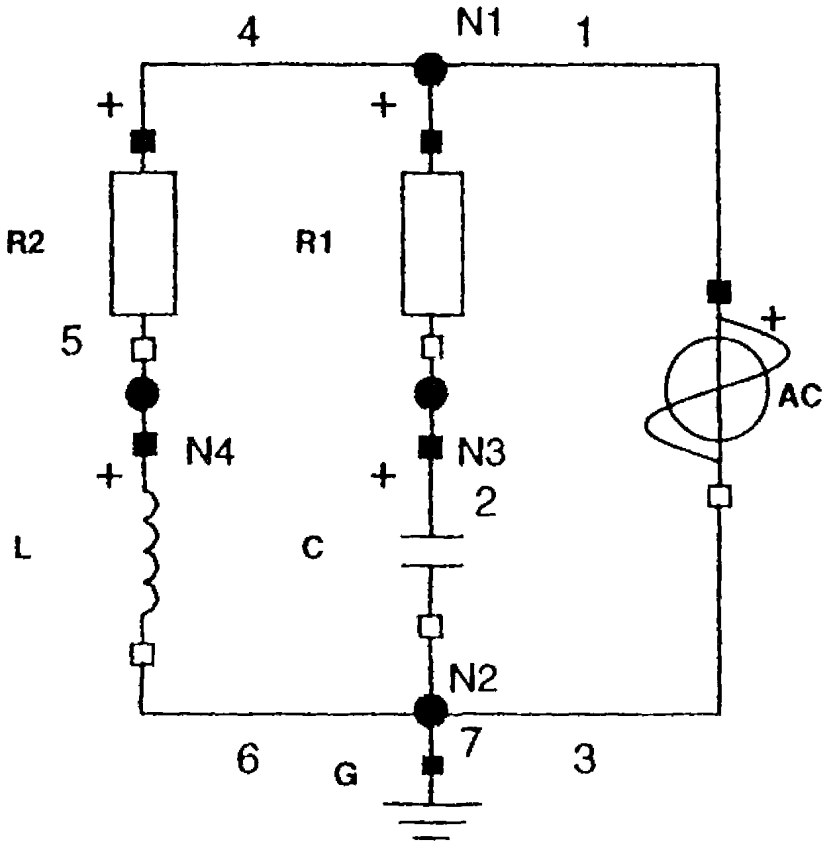
Modelica – překlad a výpočet modelu

26. června 2015

Příklad – RLC obvod

RLC obvod:




```
model Resistor
  ...
  extends Modelica.Elec...
equation
...
  v = R*i;
end Resistor;
```

Příklad – rovnice

diferenciální r:

$$der(L.i) * L.L - L.v = 0$$
$$der(C.v) * C.C - C.i = 0$$

algebraické r:

$$C.v - R1.p.v + R1.v = 0$$
$$C.i - R1.v/R1.R = 0$$
$$u(time) - R1.p.v = 0$$
$$R1.p.v - R2.v - L.v = 0$$
$$R2.v - R2.R * L.i = 0$$

- vstup, parametry, stavy, algebraické, derivace
- stejný počet stavových/algebraických proměnných jako diferenciálních/algebraických rovnic

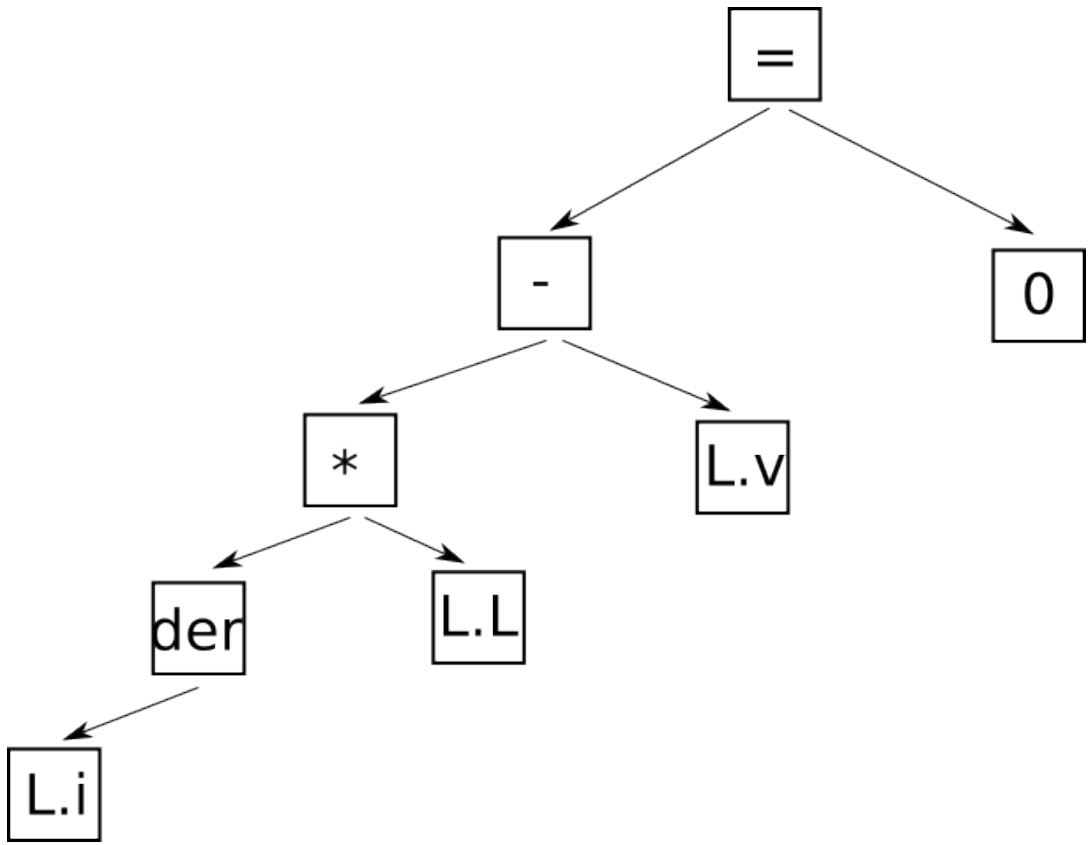
Překlad modelu – AST

1. Vytvoření AST (abstract syntax tree) modelu (parsování textových rovnic)
2. Flatenizace modelu – hierarchie objektů nahrazena jedinou soustavou rovnic (viz. předchozí příklad)
3. Analyzer – algebraické manipulace – zjednodušení rovnic, BLT transformace
4. Generování kódu v C, slinkování se solverem
5. Numerický výpočet

AST

- stromová reprezentace výrazů:

$$der(L.i) * L.L - L.v = 0$$



Flatenizace

- odstranění tříd a dědičností
- všechny rovnice do jednoho modelu
- instanční proměnné přejmenovány: `instance.variable` např. $L.i$
- odstranění hi-level modelikových konstruktů

Analyzer – BLT transformace

	$der(L.i)$	$der(C.v)$	$L.v$	$C.i$	$R1.p.v$	$R1.v$	$R2.v$
$der(L.i) * L.L - L.v = 0$	**1**	0	**1**	0	0	0	0
$der(C.v) * C.C - C.i = 0$	0	**1**	0	**1**	0	0	0
$C.v - R1.p.v + R1.v = 0$	0	0	0	0	**1**	**1**	0
$C.i - R1.v/R1.R = 0$	0	0	0	**1**	0	**1**	0
$u(time) - R1.p.v = 0$	0	0	0	0	**1**	0	0
$R1.p.v - R2.v - L.v = 0$	0	0	**1**	0	**1**	0	**1**
$R2.v - R2.R * L.i = 0$	0	0	0	0	0	0	**1**

- Incidenční matice - jednička, pokud rovnice obsahuje proměnnou
- Známe: parametry, vstupy, stavy
- Potřebuji spočítat: algebraické (abych mohl spočítat derivace), derivace (abych mohl udělat časový krok - spočítat změnu stavů)
- Prohazováním řádků a sloupců se snažíme převést na dolní trojúhelníkovou - BLT (Block Lower Triangular) transformace

Transformovaná incidenční matice – vnáší kauzalitu

	$R2.v$	$R1.p.v$	$L.v$	$R1.v$	$C.i$	$der(L.i)$	$der(C.v)$
$R2.v - R2.R * L.i = 0$	**1**	0	0	0	0	0	0
$u(time) - R1.p.v = 0$	0	**1**	0	0	0	0	0
$R1.p.v - R2.v - L.v = 0$	**1**	**1**	**1**	0	0	0	0
$C.v - R1.p.v + R1.v = 0$	0	**1**	0	**1**	0	0	0
$C.i - R1.v/R1.R = 0$	0	0	0	**1**	**1**	0	0
$der(L.i) * L.L - L.v = 0$	0	0	**1**	0	0	**1**	0
$der(C.v) * C.C - C.i = 0$	0	0	0	0	**1**	0	**1**

- Vzniká kauzalita:
  - Rovnice vyhodnocujeme odshora dolů
  - Proměnné vlevo od diagonály známe z vyhodnocení předchozích rovnic - vstupy
  - Proměnná na diagonále je z rovnice určena - výstup
- Algebraické úpravy a zjednodušení rovnic

Strong-componenty

Převod na LT není vždy možný, proto BLT – např. (jen algebraický systém)

$$
\begin{array}{rl|cccc}
 & & a & b & c & d \\
\hline
b+c+d & = -1 & 0 & 1 & 1 & 1 \\
b-c & = 2 & 0 & 1 & 1 & 0 \\
d & = 1 & 0 & 0 & 0 & 1 \\
a+ \quad d & = -3 & 1 & 0 & 0 & 1
\end{array}
$$

převedu na

$$
\begin{array}{rl|cccc}
 & & \mathrm{d} & \mathrm{b} & \mathrm{c} & \mathrm{a} \\
\hline
d & = 1 & 1 & 0 & 0 & 0 \\
d+c+b & = -1 & 1 & \mathbf{1} & \mathbf{1} & 0 \\
-c+b & = 2 & 1 & \mathbf{1} & \mathbf{1} & 0 \\
d \quad +a & = -3 & 1 & 0 & 0 & 1
\end{array}
$$

2. a 3. rovnice – systém rovnic – není možný rozřešit prohazováním řádků a sloupců - *SC* - řeší se numericky v každé iteraci výpočtu (náročné).
BLT transformace $\rightarrow$ snaha vytvořit co nejmenší strongcomponenty.
Boolovské proměnné ve strong-componentě - špatně se řeší.

Generování a překlad kódu

- generování C souboru modelu – funkce reprezentující model
- knihovna numerického solveru
- překlad $\rightarrow$ spustitelná aplikace počítá model

Numerický výpočet (Eulerova metoda)

$$\bar{x} \rightarrow \boxed{\text{vyhodnocení rovnic, řešení strong-component}} \rightarrow \dot{\bar{x}}$$

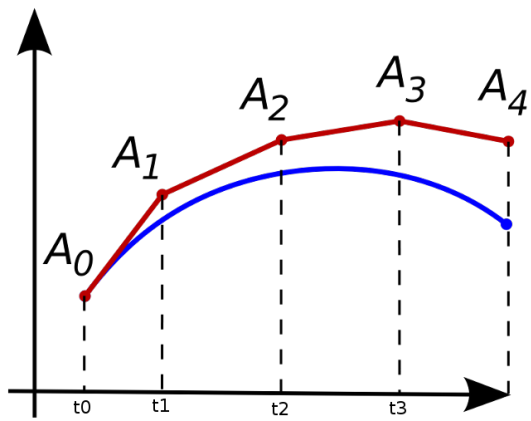
$$\dot{\bar{x}} = \bar{f}(\bar{x}, t)$$

Diskretizace proměnných a rovnice (náhrada derivace diferencí)

$$\frac{x_{n+1} - x_n}{\Delta t} = f(x_n, t_n)$$

Počítáme iterativně

$$x_{n+1} = x_n + \Delta t \cdot f(x_n, t_n)$$



Nastavení solveru

- Numerické metody (OM)
  - dassl – defaultní, robustní, pomalá (vlastní volba kroku, kontrola chyby)
  - euler – nejjednodušší, rychlý, málo přesný, málo robustní
  - rungekutta – RK4, rychlý, přesnější než euler, málo robustní
  - radau 1–5 – implicitní RK, robustnější, pomalejší

pro urychlení výpočtu (pro snadno řešitelné modely) – rungekutta

- počet intervalů
  - $\Delta t \sim 1/N$, čím více, tím přesnější (většinou)
  - pro dassl jen počet výpisů
- tolerance
  - pokud by měla být chyba přesná, dassl zkrátit krok, nebo přepne na přesnější metodu
  - ostatní metody neumějí

Hybridní systémy - podmíněné výrazy a rovnice

Volný pád ve vzduchu a vodě. Vodní hladina $x = 0$:

$$
\begin{aligned}
inAir &= x > 0 \\
f &= -gm - (\text{if } inAir \text{ then } k_1 v \text{ else } k_2 |v| v) \\
\dot{v} &= f/m \\
\dot{x} &= v
\end{aligned}
$$

- Zero-crossing funkce $g(t) = x$
- Continuous time
  - řeší se diferenciální a algebraické rovnice a postupuje se v čase
  - nenastávají události, jsme jen v jedné větvi podmínky
  - sleduje se, jestli g nemění znaménko, událost $\rightarrow$přepnutí na
- discrete time
  - zpracovává se událost
    - ★ přesná detekce času události
    - ★ řeší se dohromady spojité i diskrétní rovnice (událost může vyvolat další, hledají se hodnoty diskrétních proměnných vyhovující rovnicím)
  - čase se nemění

Výpočet DAE systému – rekapitulace

- Při překladu – tzn. jen jednou
  - Algebraické úpravy rovnic a BLT transformace
- V každé iteraci výpočtu
  - vyhodnocování rovnic – $\bar{x} \rightarrow \dot{\bar{x}}$ – numerické řešení strong-component
    - ⋆ strong-componenty mohou být nelineární, veliké (pro nelineární $N \gtrsim 10$) $\Rightarrow$**může být časově velice náročné** $\Rightarrow$u velkých modelů **potřeba strongcomponenty minimalizovat**
  - výpočet nových stavů
  - detekce zero-crossingů
  - případně zpracování události

Inicializace

Před simulací probíhá inicializace - přiřazuje se hodnota všem proměnným (stavovým, derivacím, algebraickým, parametrům).
atribut `fixed`

- `true` - hodnota proměnné je neměnná
- `false` - proměnná vystupuje v inicializaci jako neznámá
- defaultně: parametry - `true`, stavy, algebraické (,derivace) - `false`

atribut `start`

- `fixed=true` - pevná počáteční hodnota
- `fixed=false` - počáteční odhad

sekce `initialEquation`

- rovnice, které mají být při inicializaci navíc splněny společně s rovnicemi modelu

Inicializace 2

Řeší se dohromady rovnice modelu a iniciální rovnice

$$F(x, \dot{x}, y, t) = 0$$
$$G(x, y, t) = 0$$
$$I(x, \dot{x}, y, t) = 0$$

Počet všech rovnic by měl odpovídat počtu „nonFixed" proměnných(+derivací). Pokud rovnic méně, nastaví se některé proměnné na fixed.

Inicializace - příklad

```
model InitExample
  Real x(start = 1, fixed = true);
  Real y(start = 2); //fixed = false by default
initial equation
  der(y) = 0;
equation
  der(x) = x + y;
  der(y) = x - y);
end InitExample
```

Reziduální funkce

$$r_1 = \dot{y}$$
$$r_2 = \dot{x} - x - y$$
$$r_3 = \dot{y} - x + y$$

x je zadána napevno, hledáme $y, \dot{x}, \dot{y}$. Stejně rovnic jako neznámých.

$$\min_{(y,\dot{x},\dot{y})}(\dot{y}^2 + (\dot{x} - x - y)^2 + (\dot{y} - x + y)^2)$$

Materiály

- Specifikace Modeliky - www.modelica.org/documents
- Principles of Object-oriented modeling and simulation with Modelica (Peter Fritzson)
- články
  - Event Handling in the OpenModelica Compiler and Runtime System (Håkan Lundvall, Peter Fritzson, Bernhard Bachmann)
  - Dynamic Selection of States in Dymola (Sven Erik Mattsson, Hans Olsson and Hilding Elmqvist)
- Přeložené modely, zdrojový kód OpenModeliky