Tworzenie języków specyfikacji dla zagadnień numerycznych

prof. dr hab. inż. Norbert Sczygiol
dr inż. Andrzej Grosser

Instytut Informatyki Teoretycznej i Stosowanej
Politechnika Częstochowska

11 września 2012

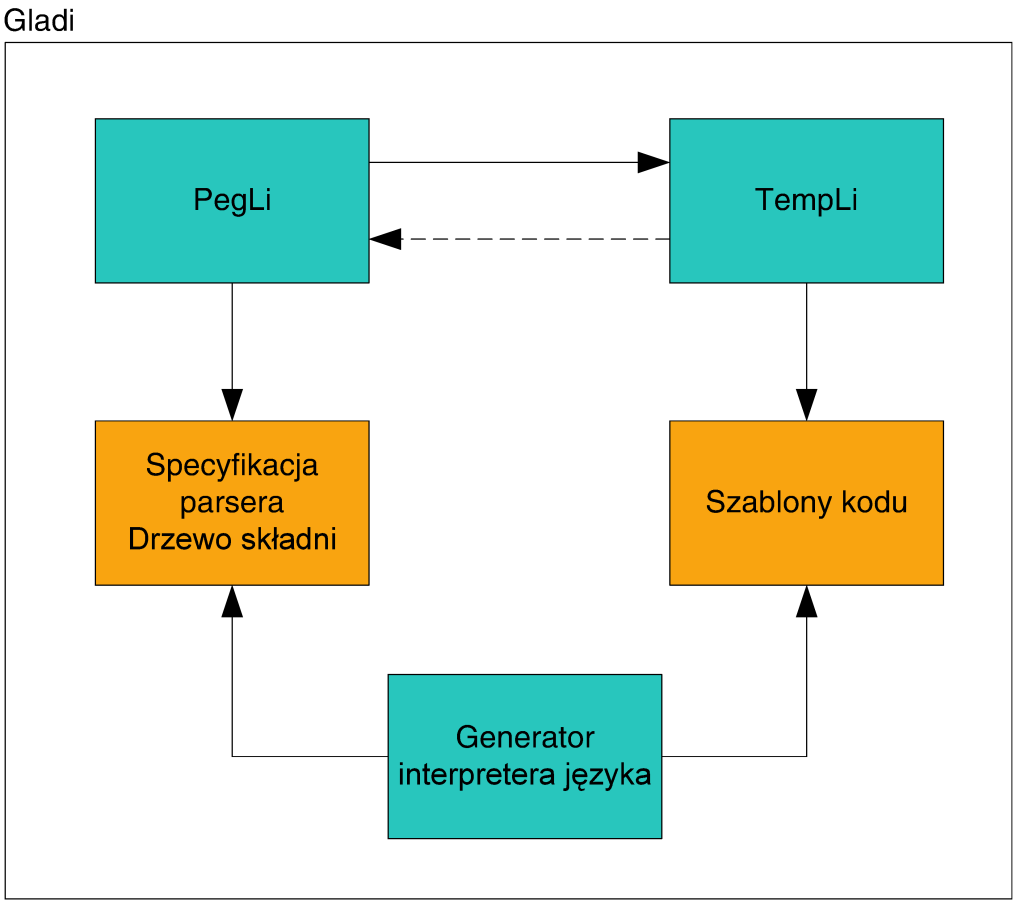
Plan prezentacji

Język specyfikacji dziedzinowej

DSL

- Języki specyfikacji dziedzinowej (DSL - *Domain Specific Language*) są językami programowania zaprojektowanymi do używania dla określonego zbioru zadań. Są one również nazywane: małymi językami, makrami, językami aplikacji, językami bardzo wysokiego poziomu.
- Ze względu na sposób ich tworzenia można je podzielić na:
  - wewnętrzne (osadzone),
  - zewnętrzne.

Środowisko do tworzenia języków specyfikacji dziedzinowej



Rysunek : Środowisko do tworzenia języków specyfikacji dziedzinowej

PegLi

- PegLi - jest biblioteka i generatorem analizatorów składniowych.
- Ta część środowiska została napisana w języku Python.
- Do opisu reguł języka wykorzystuje gramatyki wyrażeń parsujących.

Wyrażenia parsujące

Gramatyka wyrażeń parsujących

- Każda reguła jest parą w postaci (A, e), zapisywana zwykle w formie $A \leftarrow e$, gdzie A jest definiowaną reguła, natomiast e jest wyrażeniem parsującym.
- Wyrażenia parsujące definiuje się w postaci (e, e_1 oraz e_2 są wyrażeniami parsującymi):
  1. $'str'$ - dopasowanie łańcucha znaków,
  2. . - dopasowanie dowolnego znaku,
  3. $e_1 e_2$ - sekwencja,
  4. e_1 / e_2 - uporządkowany wybór,
  5. e^* - zero lub więcej powtórzeń,
  6. e^+ - jedno lub więcej powtórzeń,
  7. $!e$ - operator negacji.

PegLi

- Parser opisywany jest pomocą dwóch sekcji.
- Pierwsza (opcjonalna) sekcja dołącza gramatykę zdefiniowaną wcześniej, którą można użyć do opisu składni tworzonego języka (częściowo lub w całości).
- Kolejna sekcja jest opisem reguł. Składnia używana w niej pokrywa się z omówioną składnią wyrażeń parsujących. Jedynym rozszerzeniem jest możliwość użycia reguł zdefiniowanych w gramatykach zewnętrznych, umożliwienie definiowania akcji semantycznych i instrukcje związane z budową abstrakcyjnego drzewa składni.

TempLi

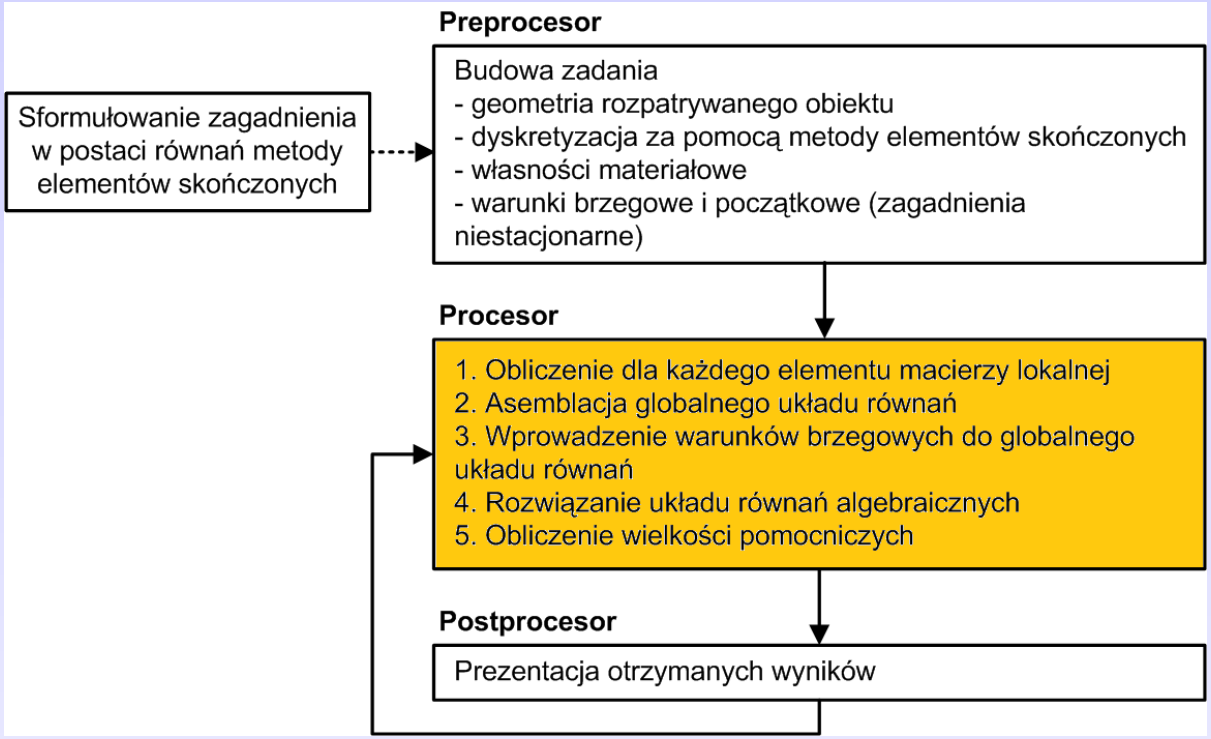
- TempLi (skrót od **Templ**ate **Li**brary) jest autorskim językiem specyfikacji dziedzinowej oraz biblioteką oprogramowania stworzoną do generowania tekstu z wykorzystaniem zewnętrznych struktur danych.
- Szablony kodu są transformowane do kodu języka Pyton
- Szablony kodu biblioteki TempLi składają się z dwóch sekcji:

```
// I sekcja - szablonów i gramatyk
using template nazwa;
using grammar gramatyka;
// II sekcja - atrybutów i wzorców generacyjnych
//deklaracja atrybutu
$x
//definicja wzorca generacyjnego
$zmienna($typ, $nazwa) <- $typ $nazwa';' $end
```

Gladi

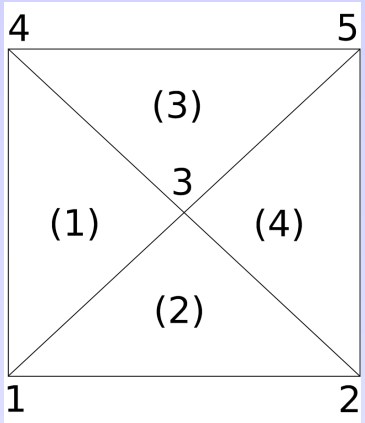
- Środowisko programowania Gladi zapewnia podświetlanie składni zarówno dla szablonów jak i dla specyfikacji parsera, generację interpreterów, wsparcie dla operacji edycyjnych.
- Zostało zaimplementowane w języku C++ z wykorzystaniem biblioteki Qt.
- Możliwe jest rozszerzanie tego środowiska za posrednictwem systemu pluginów biblioteki Qt.

Etapy metody elementów skończonych



Rysunek : Etapy uzyskiwania rozwiązania za pomocą metody elementów skończonych (zagadnienia liniowe)

Asemblacja macierzy globalnej



$$\overset{(1)}{\begin{bmatrix} \star & 0 & \star & \star & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \star & 0 & \star & \star & 0 \\ \star & 0 & \star & \star & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}} + \dots + \overset{(4)}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \star & \star & 0 & \star \\ 0 & \star & \star & 0 & \star \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \star & \star & 0 & \star \end{bmatrix}} = \overset{\Sigma}{\begin{bmatrix} \star & \star & \star & \star & 0 \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star \end{bmatrix}}$$

Rysunek : Asemblacja globalnego układu równań z macierzy lokalnych (gwiazdkami oznaczono miejsca wprowadzenia współczynników do globalnego układu równań)

Języki specyfikacji dla MES

- Języki osadzone:
  - PIER - Common Lisp,
  - Analysa - Scheme,
  - MyFem - Python,
  - Feel++ - C++.
- Języki zewnętrzne:
  - FreeFEM,
  - GetDP.

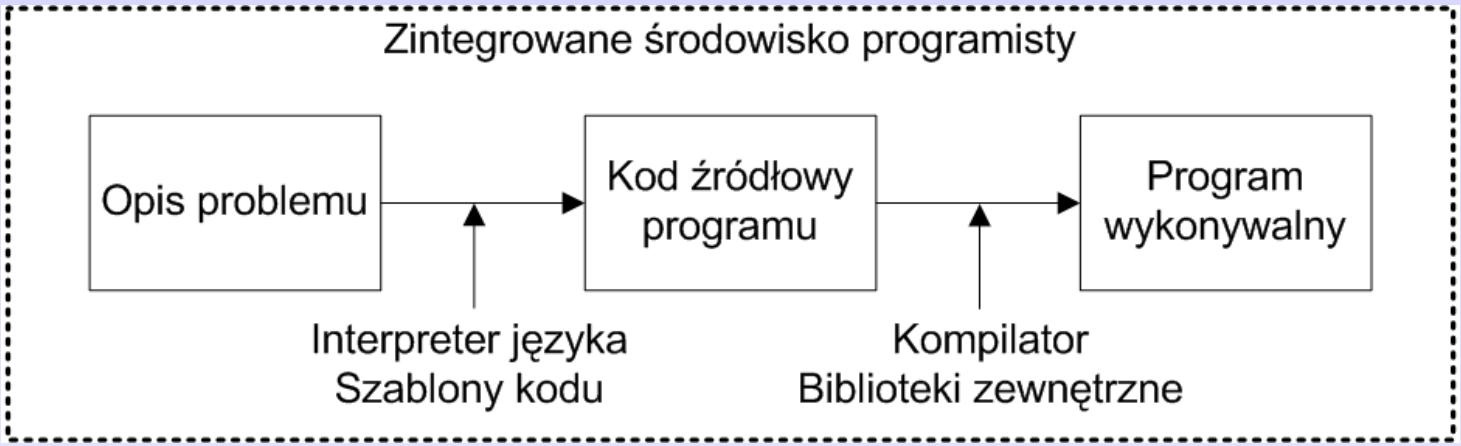
Język specyfikacji dziedzinowej - Finele

- Język specyfikacji `Finele` został zaprojektowany do wspierania tworzenia zadań za pomocą metody elementów skończonych z szczególnym uwzględnieniem budowania macierzy lokalnych i wprowadzania warunków brzegowych.
- Bazuje on na spostrzeżeniu, że niektóre etapy tworzenia rozwiązania są wspólne dla różnych zastosowań (odczyt danych wejściowych, zapis wyników, asemblacja macierzy globalnej) i mogą być automatycznie wygenerowane dla potrzeb programu, bez jawnej ich implementacji.
- Język jest zaprojektowany do budowy procesorów metody elementów skończonych.

Budowa skryptu języka Finele

- Skrypt w języku Finele składa się z sekcji opisujących poszczególne składowe zadania.
- Sekcja rozpoczynająca się od słowa kluczowego `MaterialProperty` wyszczególnia wszystkie własności materiałowe stosowane do budowy zadania, które są wprowadzane za pomocą preprocesora.
- Następna część jest odpowiedzialna za opis solwera (solwer - program lub część programu realizująca obliczenia). Wymieniane są tutaj niewiadome uzyskiwane po rozwiązaniu układu równań (podsekcja niewiadomych), deklaracje pomocniczych macierzy, wektorów i wielkości skalarnych (sekcja deklaracji), a także sposób budowania macierzy lokalnej dla elementu (podsekcja budowniczego). Przyjęto konwencję macierzową.
- Ostatnia sekcja `BoundaryConditions` opisuje warunki brzegowe.

Zastosowane podejście



Rysunek : Składniki języka specyfikacji wraz z etapami generowania kodu

Przykładowe zagadnienie

Równaniem wyjściowym, dla niejawnego schematu całkowania po czasie, jest:

$$(\mathbf{M}^{\mathbf{n+1}}+\Delta t\mathbf{K}^{\mathbf{n+1}})\mathbf{T}^{\mathbf{n+1}} = \mathbf{M}^{\mathbf{n+1}}\mathbf{T}^{\mathbf{n}}+\Delta t\mathbf{b}^{\mathbf{n+1}} \tag{1}$$

Macierze $\mathbf{M}$ i $\mathbf{K}$ w tym równaniu oblicza się z zależności:

$$\mathbf{M} = \mathbf{c}\rho \int_{\mathbf{A}} \mathbf{N}^{\mathbf{T}}\mathbf{N}\ \mathbf{dA} \tag{2}$$

$$\mathbf{K} = \lambda \int_{\mathbf{A}} \nabla^{\mathbf{T}}\mathbf{N} \cdot \nabla\mathbf{N}\ \mathbf{dA} \tag{3}$$

Przykładowe zagadnienie

```
MaterialProperty{
    lambda;
    c;
    rho;
}

Solver Heat{
   unknowns -> T;
   builder:
    << c * rho >*(w Trans,N)*dx
      + dt * <lambda> * (Nabla w Trans, Nabla N)*dx> * T
          = <<c * rho> * (w Trans,N)*dx> * T[-1];
}
```

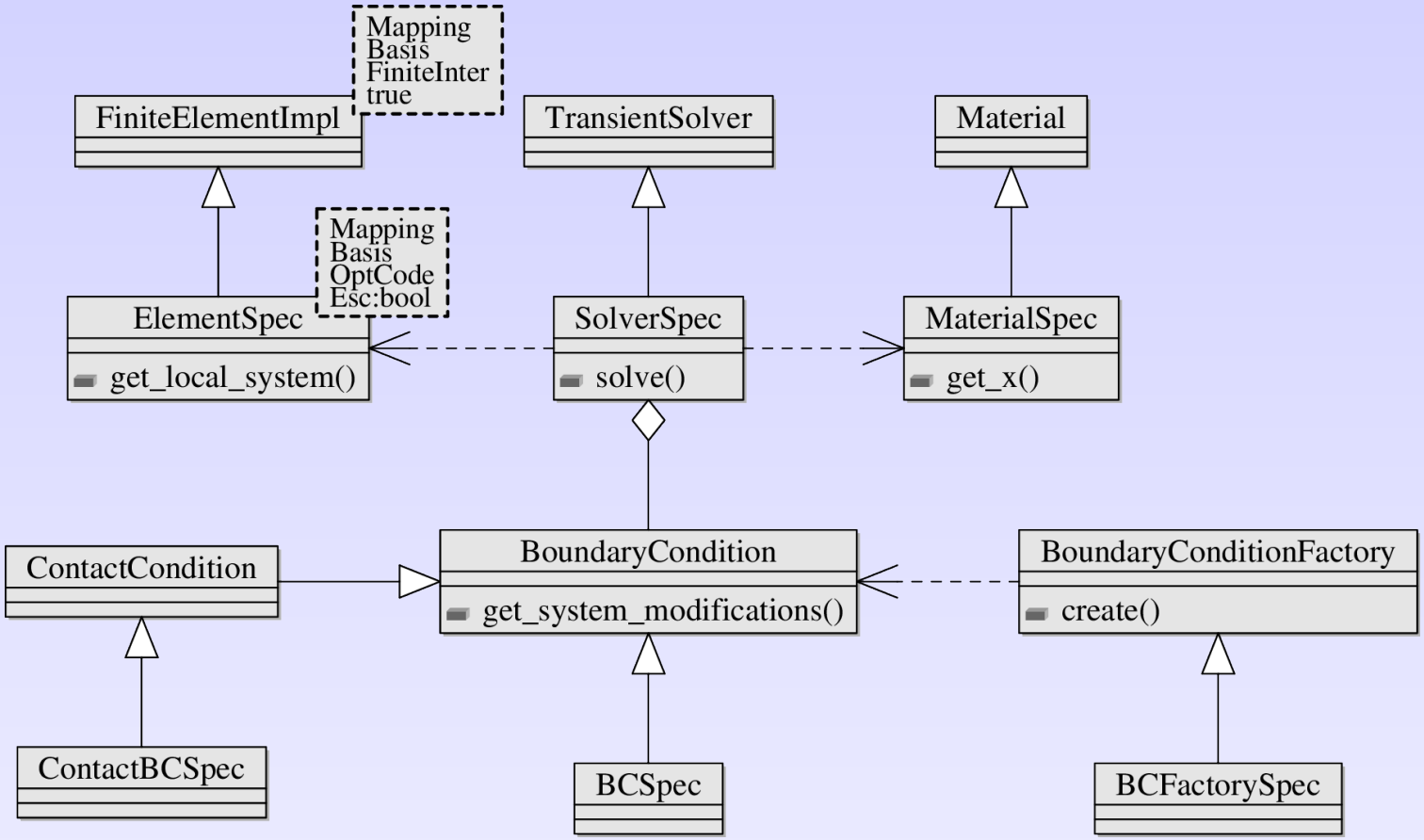
Przykładowe zagadnienie

```
BoundaryConditions{
    bc Neumann{
    coeff -> q;
    builder:
    <q>*(w Trans,N) * ds;
    }
}
```

Implementacja zagadnienia

- main.cpp - zawiera funkcję main(),
- solver.h - zawiera definicję klasy solwera,
- solver.cpp - zawiera implementacje metod solwera,
- elements.h - zawiera definicje klas opisujących elementy skończone oraz definicję ich metod,
- bcs.h - zawiera definicje klas implementujących warunki brzegowe,
- bcs.cpp - zawiera definicje metod klas warunków brzegowych,
- materials.h - zawiera definicje klas obsługujących własności materiałowe,
- materials.cpp - zawiera definicje metod klas własności materiałowych,
- Makefile - zawiera skrypt programu make.

Implementacja zagadnienia



Rysunek : Implementacja zagadnienia z wykorzystaniem biblioteki eNsc

Podsumowanie

- Praktyczne zastosowanie narzędzi i technik pokazanych w prezentacji pozwala na szybkie przygotowanie języków specyfikacji dziedzinowej.
- Wprowadzone ułatwienia obejmują zintegrowane środowisko programowania automatyzujące proces budowy języka specyfikacji dziedzinowej.
- Dzięki opartej na gramatykach wyrażeń parsujących bibliotece i generatorowi PegLi, zmniejszono trudności związane z przygotowaniem parserów dla języków specyfikacji.
- W ramach pracy stworzono również bibliotekę TempLi upraszczającą tworzenie szablonów kodu.
- Zaprezentowane narzędzia znalazły praktyczne zastosowanie przy implementacji deklaratywnego języka specyfikacji Finele - zaprojektowanym do realizacji zadań za pomocą metody elementów skończonych.

Dziękuję za uwagę