Interpretacja abstrakcyjna

Maciej Pazurkiewicz

2 grudnia 2009

O referacie

Wstęp do interpretacji abstrakcyjnej

Interpretacja abstrakcyjna to teoria poprawnego przybliżania różnych struktur matematycznych, w szczególności tych związanych z semantyką programów komputerowych.
Została sformalizowana w latach 70. XX wieku przez Patricka i Radhię Cousotów.

Abstrakcja w matematyce

$$9435192 \cdot 3562951413 + 27182818 \cdot 73947229 = 35627224735837619$$

Abstrakcja w matematyce

$$9435192 \cdot 3562951413 + 27182818 \cdot 73947229 = 35627224735837619$$

$$2 \cdot 3 + 8 \cdot 9 \neq 9 \mod 10$$

$$\Downarrow$$

$$9435192 \cdot 3562951413 + 27182818 \cdot 73947229 \neq 35627224735837619$$

Idea interpretacji abstrakcyjnej

- Nie trzeba znać wszystkich wszystkich szczegółów, aby móc stwierdzić coś o prawdziwości zdania bądź poprawności programu.
- Pozbycie się niektórych informacji zmniejsza rozmiar danych do obliczeń, co może pozwolić na udzielenie odpowiedzi w rozsądnym czasie.

Obliczanie znaku wyrażenia

$$35853333 \cdot 89333323 \cdot (-112525334 + (-9432514) + (-9233522)) < 0$$

$$(3523523 + (-452523523) + (-125325235) + 5235236) \cdot 23539 \cdot 0 = 0$$

$$2345246124062446 + (-4224691931951395) \; ? \; 0$$

Semantyka	Semantyka abstrakcyjna
Int	**Sign** $= \{+, -, 0, \top\}$
$\mathcal{E}[\![n]\!] = n$	$\mathcal{E}[\![n]\!] = \begin{cases} 0 & n = 0 \\ + & n > 0 \\ - & n < 0 \end{cases}$

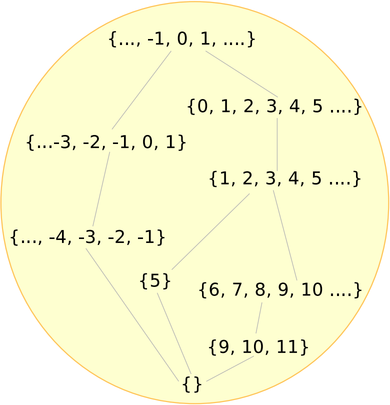
Obliczanie znaku wyrażenia

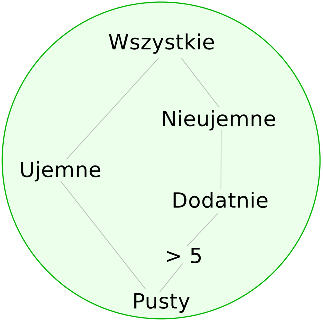
Semantyka	Semantyka abstrakcyjna
$\mathcal{E}[\![e_1 \cdot e_2]\!] = e_1 \cdot e_2$	$\mathcal{E}[\![e_1 \cdot e_2]\!] = \begin{cases} 0, & \mathcal{E}[\![e_1]\!] = 0 \vee \mathcal{E}[\![e_2]\!] = 0 \\ \top, & \mathcal{E}[\![e_1]\!] = \top \vee \mathcal{E}[\![e_2]\!] = \top \\ +, & \mathcal{E}[\![e_1]\!] = \mathcal{E}[\![e_2]\!] \\ -, & \mathcal{E}[\![e_1]\!] \neq \mathcal{E}[\![e_2]\!] \end{cases}$
$\mathcal{E}[\![e_1 + e_2]\!] = e_1 + e_2$	$\mathcal{E}[\![e_1 + e_2]\!] = \begin{cases} \mathcal{E}[\![e_1]\!], & \mathcal{E}[\![e_2]\!] = 0 \\ \mathcal{E}[\![e_2]\!], & \mathcal{E}[\![e_1]\!] = 0 \\ \top & \mathcal{E}[\![e_1]\!] \neq \mathcal{E}[\![e_2]\!] \\ \mathcal{E}[\![e_1]\!], & \mathcal{E}[\![e_1]\!] = \mathcal{E}[\![e_1]\!] \end{cases}$

Konstrukcja interpretacji abstrakcyjnej

Pytanie	**Odpowiedź**
Które własności mnie interesują?	określenie dziedziny abstrakcyjnej
Jaka jest interpretacja abstrakcyjna podstawowych elementów dziedziny?	określenie funkcji abstrakcji
Jak działać na wartościach abstrakcyjnych?	definicja semantyki abstrakcyjnej operacji

Porządek na dziedzinach





Dziedzina $D = \mathcal{P}(\mathbf{Int})$ z porządkiem $\subseteq$.

Dziedzina abstrakcyjna D_{abs} z porządkiem $\sqsubseteq$.

Porządek na dziedzinach

Dziedzina i dziedzina abstrakcyjna to kraty zupełne.

Definicja (Krata zupełna)

Krata zupełna $\langle P, \sqsubseteq, \sqcup \rangle$ *to taki porządek częściowy* $\langle P, \sqsubseteq \rangle$*, w którym każdy podzbiór elementów* $X \subseteq P$ *ma kres dolny* $\sqcup X$*.*

Przykłady:

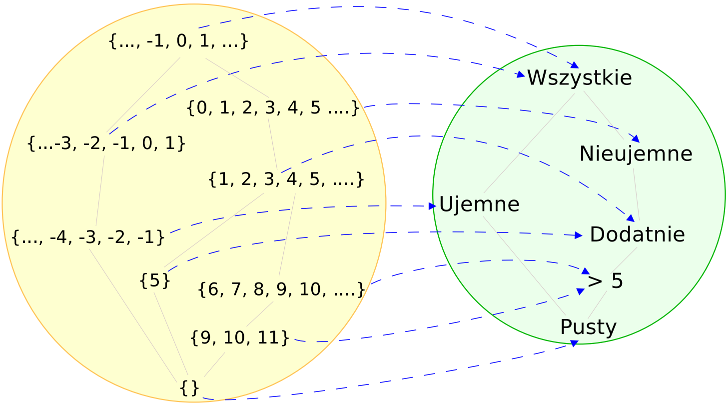
- zbiór potęgowy danego zbioru uporządkowany przez inkluzję
- przedział domknięty liczb rzeczywistych ze zwykłym porządkiem

Kraty to porządne zbiory

Krata zupełna L ma *dobre* własności:

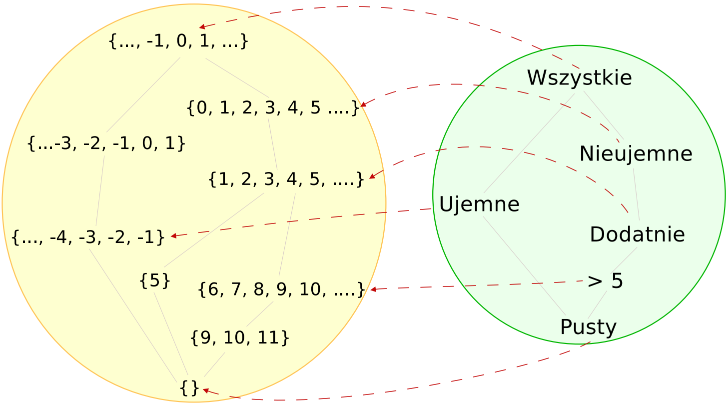
- ma element najmniejszy
- ma element największy
- jest niepusta
- jest porządna jeśli chodzi o punkty stałe.
  - **Tw. Knastera-Tarskiego**. Jeśli L jest kratą i $F : L \longrightarrow L$ jest monotoniczna to F ma najmniejszy punkt stały. Co więcej, zbiór wszystkich punktów stałych F jest także kratą zupełną.

Funkcja abstrakcji



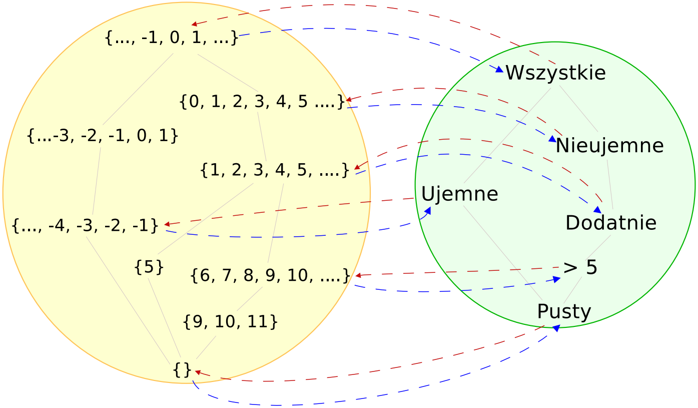
Funkcja abstrakcji $\alpha : D \longrightarrow D_{abs}$ przyporządkowuje zbiorowi danych wartość abstrakcyjną, która najlepiej go opisuje.

Funkcja konkretyzacji



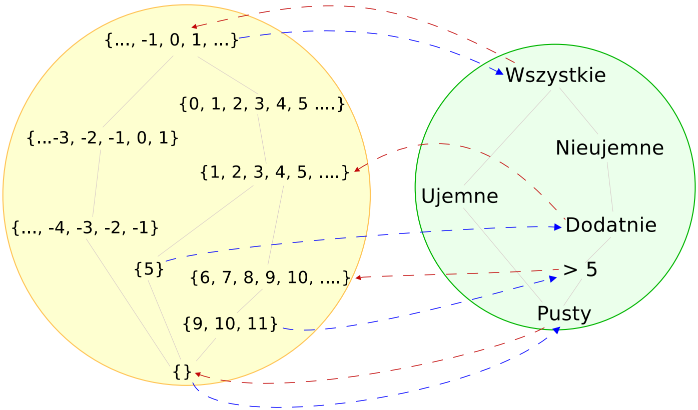
Funkcja konkretyzacji $\gamma : D_{abs} \longrightarrow D$ przypisuje wartości abstrakcyjnej reprezentowany przez nią zbiór danych.

Abstrakcja złożona z konkretyzacją



$$\alpha \circ \gamma(a) = a$$

Konkretyzacja złożona z abstrakcją



$$X \subseteq \gamma \circ \alpha(X)$$

Własności funkcji abstrakcji i konkretyzacji

Chcielibyśmy, żeby funkcja abstrakcji α oraz funkcja konkretyzacji γ miały następujące własności:

- monotoniczność
  - $X \subseteq Y \Longrightarrow \alpha(X) \sqsubseteq \alpha(Y)$
  - $a \sqsubseteq b \Longrightarrow \gamma(a) \subseteq \gamma(b)$
- $X \subseteq \gamma \circ \alpha(X)$
- $\alpha \circ \gamma(a) \sqsubseteq a$
  - A nawet: $a = \alpha \circ \gamma(a)$, co oznacza, że wartości abstrakcyjne są dokładne.

(X, Y - zbiory danych rzeczywistych; a, b - wartości abstrakcyjne)

Połączenie Galois

$$\begin{cases} \text{motoniczność } \alpha \text{ i } \gamma \\ X \subseteq \gamma \circ \alpha(X) \\ \alpha \circ \gamma(a) \sqsubseteq a \end{cases} \equiv \quad X \subseteq \gamma(a) \iff \alpha(X) \sqsubseteq a$$

Są to cechy połączenia Galois(*Galois connection*).

- ma zastosowanie w różnych działach matematyki i informatyki teoretycznej
- w interpretacji abstrakcyjnej oznacza parę funkcji będącą najlepszą parą funkcji abstrakcji i konkretyzacji

Podsumowanie

- Podejście bardzo naturalne, choć podbudowane poważną teorią matematyczną.
- Liczymy na to, że pewne własności da się wykazać, pomimo (a często dzięki!) pozbycia się pewnych szczegółów.
- Mamy pełność i poprawność - w takim sensie, że sytuacja błędna nie zostanie pominięta, ani nie otrzymamy fałszywego alarmu. Możemy jednak uzyskać odpowiedź “nie wiem”.

Przegląd zastosowań

- Weryfikacja programów
  - najlepiej działa tam, gdzie badamy własności zmiennych liczbowych
  - gorzej dla dużych wskaźnikowych struktur danych
- Analiza złożonych cech programu
  - analiza ścisłości (*strictness analysis*) - funkcja f jest ścisła, gdy $f(\perp) = \perp$
  - analiza żywych zmiennych (*live variable analysis*) - zmienna jest żywa, gdy jej wartość może być użyta w którejkolwiek ścieżce programu, zanim zostanie jej przypisana nowa wartość
- Optymalizacja w trakcie kompilacji
- Analiza wykorzystania zasobów

Wstęp do analizy pięciokątów

Analiza pięciokątów

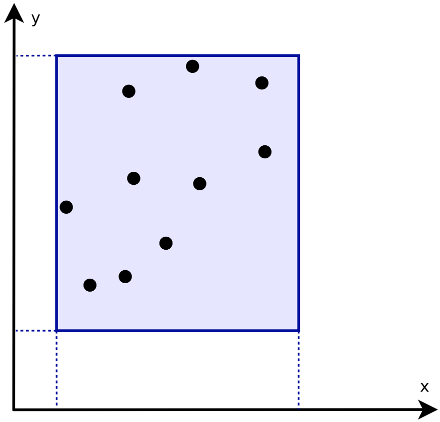
- przykład metody statycznej weryfikacji programów opartej na interpretacji abstrakcyjnej
- zaimplementowana jako część *Clousot* - analizatora dla programów w .NET

Abstrakcyjne dziedziny liczbowe

Abstrakcyjne dziedziny liczbowe:

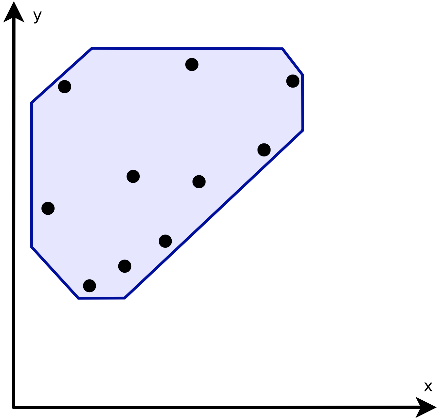
- abstrakcje dla zestawów konkretnych wartości zmiennych
- zamiast przechowywać wartość zmiennej, pamiętamy tylko pewien zbiór, do którego należy
- wykorzystywane w weryfikacji programów pod kątem występowania prostych błędów czasu wykonania (dzielenie przez zero, przekroczenie zakresu tablicy itp.)

Przedziały



Dziedzina przedziałów.
Ograniczenia postaci: $v \in [a, b]$

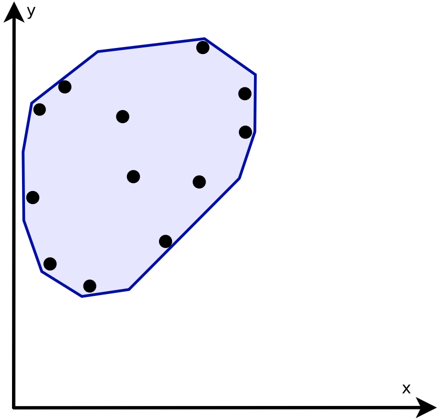
Ośmiokąty



Dziedzina ośmiokątów.

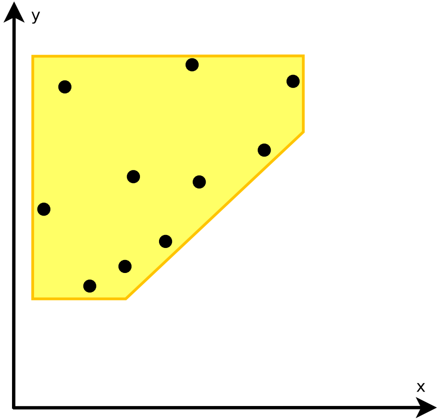
Ograniczenia postaci: $\pm x \pm y \leq k$

Wielościany wypukłe



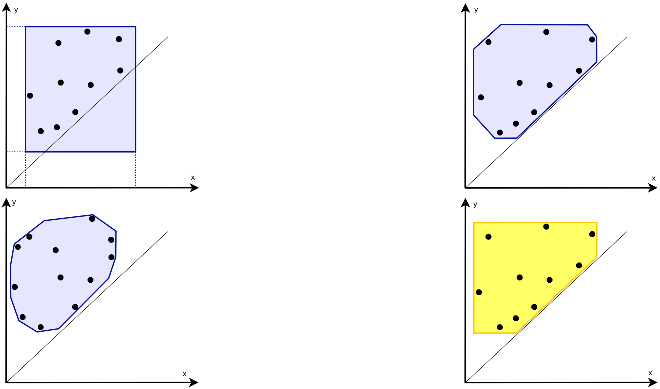
Dziedzina wielościanów wypukłych.
Ograniczenia postaci:
$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \leq k$

Pięciokąty



Dziedzina pięciokątów.
Ograniczenia postaci:
$x \in [a, b] \wedge x < y$

Pięciokąty - motywacja



Pięciokąty zachowują minimalną ilość informacji do wywnioskowania niezmienników typu $0 \leq \text{index} \leq \text{array.length}$

Przedziały

Najprostsza abstrakcyjna dziedzina liczbowa Intv. Pamiętamy minimum i maksimum wartości zmiennej.

Definicja

Porządek $[a_1, b_1] \sqsubseteq_i [a_2, b_2] \iff a_1 \geq a_2 \wedge b_1 \leq b_2$

Najmniejszy element $[a, b] = \bot_i \iff a > b$

Największy element $[a, b] = \top_i \iff a = -\infty \wedge b = \infty$

Suma $[a_1, b_1] \sqcup_i [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)]$

Iloczyn $[a_1, b_1] \sqcap_i [a_2, b_2] = [\max(a_1, a_2), \min(b_1, b_2)]$

Przedziały

Po funkcyjnym podniesieniu otrzymujemy środowiska przedziałów:
$\text{Boxes} = [\text{Var} \rightarrow \text{Intv}]$

Definicja

Porządek	$b_1 \sqsubseteq b_2 \iff \forall x \in b_1\ b_1(x) \sqsubseteq_i b_2(x)$
Najmniejszy element	$b = \bot \iff \exists x \in b\ b(x) = \bot_i$
Największy element	$b = \top \iff \forall x \in b\ b(x) = \top_i$
Suma	$b_1 \sqcup b_2 = \lambda x . b_1(x) \sqcup_i b_2(x)$
Iloczyn	$b_1 \sqcap b_2 = \lambda x . b_1(x) \sqcap_i b_2(x)$

Ograniczenia górne

Najprostsza dziedzina abstrakcyjna, która zachowuje pewne informacje o związkach między wartościami zmiennych. Każdej zmiennej przypisujemy zbiór zmiennych większych od niej.

$$x \mapsto \{y_1, y_2, y_3, \ldots\} \Longrightarrow \forall i\ y_i > x$$

Definicja

$$\begin{aligned}
\textit{Porządek} \quad & s_1 \sqsubseteq s_2 \iff \forall x \in s_2\ s_1(x) \supseteq s_2(x) \\
\textit{Najmniejszy element} \quad & s = \bot \iff \exists x, y \in s\ x \in s(y) \land y \in s(x) \\
\textit{Największy element} \quad & s = \top \iff \forall x \in s\ s(x) = \emptyset \\
\textit{Suma} \quad & s_1 \sqcup s_2 = \lambda x. s_1(x) \cap s_2(x) \\
\textit{Iloczyn} \quad & s_1 \sqcap s_2 = \lambda x. s_1(x) \cup s_2(x)
\end{aligned}$$

Pięciokąty

Można by spróbować połączyć obie dziedziny środowisk przedziałów oraz ograniczeń górnych.

- Podejście pierwsze: utworzyć dziedzinę będącą iloczynem kartezjańskim. Jest to równoważne prowadzeniu dwóch analiz jednocześnie bez żadnej wymiany informacji.
- Podejście drugie: skorzystać ze zredukowanego iloczynu kartezjańskiego: nie rozróżniać par $\langle b, s \rangle$, które mają prawie to samo znaczenie.
  - Pozwoli to lepiej wykorzystać informacje liczbowe trzymane w Intv i symboliczne w Sub

Pięciokąty

$$\mathrm{Pntg} = \{\langle b, s\rangle \mid b \in \mathrm{Boxes}, s \in \mathrm{Sub}\}$$

Definicja

Porządek $\langle b_1, s_1\rangle \sqsubseteq \langle b_2, s_2\rangle \iff b_1 \sqsubseteq_b b_2 \wedge$
$\forall x \in s_2 \forall y \in s_2(x) \;\; y \in s_1(x) \vee (\sup b_1(x) < \inf b_1(y))$

Najmniejszy $\langle b, s\rangle = \bot \iff b = \bot_b \vee s = \bot_s$

Największy $\langle b, s\rangle = \top \iff b = \top_b \wedge s = \top_s$

Iloczyn $\langle b_1, s_1\rangle \sqcap \langle b_2, s_2\rangle = \langle b_1 \sqcap_b b_2, s_1 \sqcap_s s_2\rangle$

Pięciokąty

Co z sumą dwóch pięciokątów?

Definicja

$$\langle b_1, s_1\rangle \sqcup \langle b_2, s_2\rangle =$$
$$\quad \text{let } b^{\sqcup} = b_1 \sqcup_b b_2 \text{ in}$$
$$\quad \text{let } s^{\sqcup} = \lambda x. s'(x) \cup s''(x) \cup s'''(x)$$
$$\quad\quad \text{where } s' = \lambda x. s_1(x) \cap s_2(x)$$
$$\quad\quad \text{and } s'' = \lambda x. \{y \in s_1(x) | \sup b_2(x) < \inf b_2(y)\}$$
$$\quad\quad \text{and } s''' = \lambda x. \{y \in s_2(x) | \sup b_1(x) < \inf b_1(y)\}$$
$$\quad \text{in} \langle b^{\sqcup}, s^{\sqcup}\rangle$$

Przykłady semantyki

- $[\![x == y]\!]\langle b, s\rangle = \langle b[x, y \mapsto b(x) \sqcap_i b(y)], s[x, y \mapsto s(x) \cup s(y)]\rangle$
- $[\![x := y]\!]\langle b, s\rangle = \langle b[x \mapsto b(y)], s[x \mapsto (s(y) \setminus \{x\})]\rangle$
- $[\![\mathbf{if}(v)\ x := a\ \ \mathbf{else}\ \ x := b]\!]p = ([\![x := a]\!]p) \sqcup_p ([\![x := b]\!]p)$, o ile wartość v jest nieznana
- $[\![r := x - y]\!]\langle b, s\rangle =$
  $\langle b[r \mapsto (b(x) \ominus b(y)) \sqcap_i (x \in s(y)\ ?\ [1, \infty]\ :\ \top_i)],$
  $s[r \mapsto \inf b(y) > 0\ ?\ x \cup s(x) : \emptyset]\rangle$

Suma pięciokątów

```
if (...)
    x := 0; y := 3; // p1
else
    x := −2; y := 1; // p2
// p3
assert(x < y);
```

$$p_1 = \langle b_1, s_1 \rangle = \langle b[x \mapsto [0,0], y \mapsto [3,3]], s \rangle$$
$$p_2 = \langle b_2, s_2 \rangle = \langle b[x \mapsto [-2,-2], y \mapsto [1,1]], s \rangle$$
$$p_3 = p_1 \sqcup_p p_2 = \langle b[x \mapsto [-2,0], y \mapsto [1,3]], s \rangle$$

Można stwierdzić poprawność asercji.

Suma pięciokątów

```
if (...)
    x := 0; y := 3; // p1
else
    x := -2; y := 0; // p2
// p3
assert(x < y);
```

$$p_1 = \langle b_1, s_1 \rangle = \langle b[x \mapsto [0,0], y \mapsto [3,3]], s \rangle$$
$$p_2 = \langle b_2, s_2 \rangle = \langle b[x \mapsto [-2,-2], y \mapsto [0,0]], s \rangle$$
$$p_3 = p_1 \sqcup_p p_2 = \langle b[x \mapsto [-2,0], y \mapsto [0,3]], s \rangle$$

Nie można stwierdzić poprawności asercji

Suma pięciokątów z domknięciem

Można spróbować nieco inaczej zdefiniować operację $\sqcup_p$.

Definicja (Domknięcie)

Domknięciem pięciokąta $\langle b, s\rangle$ nazywamy pięciokąt $\langle b^*, s^*\rangle$ taki, że:

$b^* = \prod_{(x<y)\in s} [\![x < y]\!] b$

$s^* = \lambda x.s(x) \cup \{y \in b | x \neq y \wedge \sup b^*(x) < \inf b^*(y)\}$

Definicja (Suma pięciokątów)

$\langle b_1, s_1\rangle \sqcup_p^* \langle b_2, s_2\rangle = \langle b_1^* \sqcup_b b_2^*, s_1^* \sqcup_s s_2^*\rangle$

Suma pięciokątów z domknięciem

```
if (...)
    x := 0; y := 3; // p1
else
    x := −2; y := 0; // p2
// p3
assert(x < y);
```

$$p_1^* = \langle b_1^*, s_1^* \rangle = \langle b[x \mapsto [0,0], y \mapsto [3,3]], s[x \mapsto \{y\}] \rangle$$
$$p_2^* = \langle b_2^*, s_2^* \rangle = \langle b[x \mapsto [-2,-2], y \mapsto [0,0]], s[x \mapsto \{y\}] \rangle$$
$$p_3 = p_1^* \sqcup_p p_2^* = \langle b[x \mapsto [-2,0], y \mapsto [0,3]], s[x \mapsto \{y\}] \rangle$$

Możemy stwierdzić poprawność asercji.

Dokładniej nie znaczy lepiej

	Pntg $\sqcup_p$		Pntg $\sqcup_p^*$		
Plik	%	Czas	%	Czas	Przekroczenie czasu
`mscorlib.dll`	83,19	3:10	82,77	10:33	1
`System.dll`	86, 80	2:36	85,30	9:43	1
`System.Web.dll`	92.02	2:55	92.11	8:30	0
`System.Design.dll`	96,35	2:21	94,56	7:39	1
Średnio	88,89		88,10		

Tabela. Zweryfikowane dostępy do tablicy przy dwóch strategiach sumowania pięciokątów

Zastosowanie domknięcia daje bardzo niewielki zysk w dokładności analizy i bardzo duży wzrost czasu jej trwania.

Podsumowanie

Konstrukcja dobrej metody analizy:

- przemyślana konstrukcja dziedziny abstrakcyjnej
- decyzje dotyczące konkretnych operacji w semantyce abstrakcyjnej podjęte na podstawie wyników testów
- dobór dodatkowych heurystyk

Podsumowanie - przyspieszanie analizy

Metody ulepszania analizy:

- weryfikacja prostych przypadków przy pomocy mniej dokładnej dziedziny, zastosowanie dokładniejszej dziedziny tylko tam, gdzie pierwsza analiza nie przyniosła skutku
- podział zmiennych na kubełki - nie łączymy zmiennych, niemających ze sobą związku
- przybliżanie punktów stałych równań
- wskazówki dla analizatora

Bibliografia:

1. Francesco Logozzo, Manuel Fähndrich – *Pentagons: A weakly relational abstract domain for the efficient validation of array accesses*
2. David Schmidt – *All the world is an abstract interpretation (of all the world)*
3. Patrick Cousot – *Abstract Interpretation:Achievements and Perspectives*
4. Mads Rosendahl – *Introduction to Abstract Interpretation*
5. Patrick Cousot – Wykład z intepretacji abstrakcyjnej na MIT `http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/`