Modelando a Determinação de Potenciais Du-Caminhos Através da Análise de Fluxo de Dados

Marcos L. Chaim†, José C. Maldonado‡, Mario Jino†

Departamento de Engenharia de Computação e Automação Industrial†,
Universidade Estadual de Campinas (UNICAMP),
C.P. 6101, 13081 Campinas, SP

Departamento de Ciência da Computação e Estatística‡,
Universidade de São Paulo (USP),
C. P. 668, 13560 São Carlos, SP

Resumo

Neste trabalho apresenta-se uma abordagem para a solução do problema da determinação dos potenciais du-caminhos estabelecido pelos critérios Potenciais Usos [MAL88a], [MAL88b]. Mostra-se que este problema constitui uma Estrutura Monotônica Distributiva de Análise de Fluxo de Dados [HEC77] e que a solução MOP [HEC77] determina o resultado esperado, viabilizando, desta forma, a utilização dos algoritmos de Horwitz, Demers e Teitelbaum [HOR87] que são otimizações dos algoritmos propostos por Kildall [HEC77] e Kam e Ullman [KAM77]. Adicionalmente, estabelece-se uma estimativa para o custo de processamento associado à determinação do conjunto de potenciais du-caminhos.

Abstract

An approach to obtain the solution of the potential du-path determination problem, established by Potential Uses Criteria [MAL88a], [MAL88b], is presented. It is shown that this problem is a distributive monotonic data flow analysis framework [HEC77] and that the MOP Solution [HEC77] solves this problem, enabling, in this way, the use of Horwitz, Demers and Teitelbaum's algorithms [HOR87] which are optimizations of Kildall [HEC77] and Kam and Ullman [KAM77] algorithms. Furthermore, an estimated processing cost of the determination of the potential du-path set is presented.

1 Introdução

Originalmente, a análise de fluxo de dados foi aplicada para a resolução de problemas de otimização de código, na construção de compiladores. Entretanto, outras aplicações para este tipo de análise têm surgido, principalmente aquelas que visam a melhoria da qualidade do software. Fosdick e Osterweil [FOS76] utilizam a análise de fluxo de dados para a detecção de anomalias em programas; Herman [HER76] cria um critério baseado

no fluxo de dados para a seleção de casos de teste.

Particularmente no tocante a teste de software, vários outros critérios de seleção de casos de teste baseados em fluxo de dados têm surgido ultimamente [LAS83], [NTA84], [RAP85], [MAL88a]. Estes critérios têm a característica comum de utilizarem a análise de fluxo de dados para a determinação de associações, em um grafo de controle, entre atribuições de valores às variáveis de programas e os usos dessas variáveis; essas associações devem ser testadas baseado na intuição de que não se tem confiança da correta atribuição de um valor a uma variável se não criarmos um caso de teste que execute um caminho do ponto de atribuição até o ponto onde o valor da variável é utilizado [FRA88].

Assim, o problema de cálculo das associações críticas de teste em um programa pode ser encarado como um problema de análise de fluxo de dados; daí a utilidade deste tipo de análise na atividade de teste.

Neste trabalho pretende-se utilizar ferramentas da análise de fluxo de dados para a determinação das associações críticas de teste para os critérios Potenciais Usos, propostos por Maldonado, Chaim e Jino em [MAL88a] e [MAL88b]. Os critérios Potenciais Usos requerem que as associações entre a atribuição de um valor a uma variável e um potencial uso dessa variável sejam testadas, demandando casos de teste que executem caminhos entre os pontos de atribuição de valores até os pontos onde existem possíveis usos dessas variáveis.

Existe uma série de algoritmos eficientes [HEC77], [HOR87], [KAM77], que resolvem os problemas de análise de fluxo de dados. Entretanto, é necessário que esses problemas estejam inseridos dentro de uma Estrutura Monotônica de Análise de Fluxo de Dados [HEC77], [KAM77], [HOR87] (abreviada por *estrutura*). Uma *estrutura* é uma abstração que procura tratar os problemas de análise de fluxo de dados através da teoria de semi-reticulados.

Neste trabalho mostra-se que o problema de determinação das associações críticas de teste para os critérios Potenciais Usos é uma *estrutura* e, em consequência, pode-se utilizar os algoritmos acima mencionados na ferramenta de apoio à utilização dos critérios Potenciais Usos, atualmente em desenvolvimento. Adicionalmente, este resultado permite estabelecer alguns limites quanto ao custo de processamento dos critérios.

A seguir, na Seção 2 são apresentados os critérios Potenciais Usos. Na Seção 3 são definidas as estruturas monotônicas de análise de fluxo de dados. Na Seção 4 apresenta-se o modelamento da determinação dos potenciais du-caminhos (essenciais para a automatização dos critérios potenciais usos) como uma *estrutura*; são também discutidas as implicações deste resultado no custo de processamento dos critérios. A Seção 5 contém as conclusões do trabalho.

2 Critérios Potenciais Usos

2.1 Conceitos Básicos

Os critérios Potenciais Usos são uma extensão da família de critérios baseada em fluxo de dados de Rapps e Weyuker [RAP85]. A terminologia e conceitos apresentados nesta seção são essencialmente uma síntese dos trabalhos de Rapps, Frankl e Weyuker [RAP85], [FRA88], acrescidos de alguns conceitos e terminologia pertinentes aos critérios Potenciais Usos.

A representação de um programa Q como um grafo de fluxo de controle consiste no estabelecimento de correspondência entre blocos do programa e nós e na indicação de possíveis fluxos de controle entre blocos através de arcos entre os nós.

Considerando um grafo de fluxo de controle $G(N, E, s)$, onde N é o conjunto de nós, E o conjunto de arcos e s o nó início, onde nó início é definido como o único nó $s \in N$ tal que não existe $(n_k, s) \in E$ para qualquer $n_k \in N$, definem-se: *caminho*, como sendo uma sequência finita de nós $(n_1, \ldots, n_k)$, $k \geq 2$, tal que existe um arco de n_i para n_{i+1}, $i = 1, 2, \ldots, k-1$; *caminho simples* , como um caminho onde todos os nós, exceto possivelmente o primeiro e o último, sejam distintos; *caminho livre de laço* , como um caminho com todos os nós distintos; e *caminho completo*, como um caminho tal que o primeiro nó é o nó início e o último nó do caminho é o nó saída, onde nó saída é o nó $n_e \in N$ tal que não existe $(n_e, n_k) \in E$ para qualquer $n_k \in N$.

Uma ocorrência de variável num programa na linguagem de programação definida por Rapps e Weyuker [RAP85] é uma *definição de variável* , se ela estiver: i) no lado esquerdo de um comando de atribuição ou ii) num comando de entrada; e é um *uso de variável* se não for uma definição.

O grafo de fluxo de controle estendido pela associação a cada nó i do conjunto de variáveis definidas em i *(defg(i))* é denominado *grafo def.*

Um caminho $(i, n_1, \ldots, n_m, j), m \geq 0$, que não contenha definição de uma variável x nos nós $n_1, \ldots, n_m$, é chamado de *caminho livre de definição com respeito a (c.r.a.) x do nó i ao nó j e do nó i ao arco (n_m, j)* .

Um caminho livre de definição c.r.a. x $(n_1, n_2, \ldots, n_j, n_k)$ do nó i ao nó n_j e do nó i ao arco (n_j, n_k) onde o caminho $(n_1, n_2, \ldots, n_j)$ é um caminho livre de laço e n_1 tem uma definição de $x, x \in defg(n_1)$, é denominado *potencial du-caminho* c.r.a. x. O conjunto $deff(n_1, n_2, \ldots, n_j)$, associado ao caminho $(n_1, n_2, \ldots, n_j)$, é definido da seguinte maneira:

$$deff(n_1, n_2, \ldots, n_j) = defg(n_1) - [\bigcup_{1<m<j} [defg(n_1) \cap defg(n_m)]],$$

e contém o conjunto das variáveis definidas em n_1 e que não foram re-definidas no caminho $(n_2, \ldots, n_{j-1})$, ou seja, estão "vivas" no nó n_j através do caminho.

2.2 Critérios Potenciais Usos

Os critérios potenciais usos – todos-potenciais-usos e todos-potenciais-du-caminhos – requerem basicamente que caminhos livres de definição, a partir de qualquer nó i que

possua definição de variável e com respeito a qualquer variável x definida em i, sejam executados, independentemente de ocorrer uso dessa variável nesses caminhos. Neste sentido, pode-se verificar, por exemplo, que o valor de x não foi alterado nesses caminhos (possivelmente devido a efeitos colaterais) ganhando-se, desta forma, maior confiança de que a computação correta é realizada; isto vem de encontro à filosofia discutida por Myers [MYE79]: "um erro está claramente presente se um programa não faz o que supõe-se que ele faça, mas erros estão também presentes se um programa faz o que supõe-se que não faça."

- *Critério Todos-potenciais-usos:* P satisfaz o critério todos-potenciais-usos se, para todo nó i e para toda variável x para a qual existe uma definição em i, P inclui pelo menos um potencial-du-caminho c.r.a. x do nó i para todo nó e para todo arco possível de ser alcançado a partir de i.
- *Critério Todos-potenciais-du-caminhos :* P satisfaz o critério todos-potenciais-du-caminhos se para todo nó i e para toda variável x para a qual existe uma definição em i, P inclui todos potenciais du-caminhos c.r.a. x em relação ao nó i.

Maldonado, Chaim e Jino [MAL88a] fazem uma análise comparativa dos critérios Potenciais Usos com a família de critérios de Rapps e Weyuker baseada na relação de inclusão (ordem parcial), introduzida em [RAP85], e na complexidade dos critérios — número de casos de teste requeridos pelo critério no pior caso. Concluiu-se que os critérios Potenciais Usos, além de incluírem os critérios de Rapps e Weyuker, exigem, no pior caso, o mesmo número de casos de testes destes, ou seja, a complexidade dos critérios todos-potenciais-usos e todos-potenciais-du-caminhos são as mesmas que as dos critérios todos-usos e todos-du-caminhos (da família de critérios de Rapps e Weyuker), respectivamente.

3 Estruturas Monotônicas de Análise de Fluxo de Dados

O objetivo da análise de fluxo de dados é produzir uma atribuição ("assignment") — fato de programa ("program fact") — associado a cada nó de um grafo de fluxo de controle $G(N, E, s)$ representando um programa, que será válido toda vez que o nó for atingido durante uma possível execução [HOR87].

As *estruturas* são um modelo para os problemas de análise de fluxo de dados. Os *fatos de programa* são elementos de um semi-reticulado L de comprimento finito, operação "meet" $\sqcap$, elemento mínimo $\perp$ e ordem parcial $\sqsubseteq$. Uma função $f : L \rightarrow L$ está associada a cada arco $e = (n_i, n_j) \in E$, tal que, se o fato x está associado a n_i e o fluxo de controle passa por e, então o fato $f(x)$ será associado a n_j.

As definições acima e as que se seguem nesta seção foram retiradas, em sua maior parte, de [HOR87].

Definição 1 *Uma função $f : L \to L$ é monotônica se, e somente se (sse), $\forall x, y \in L$, $x \sqsubseteq y$ implica $f(x) \sqsubseteq f(y)$. Equivalentemente, f é monotônica sse $\forall x, y \in L$, $f(x \sqcap y) \sqsubseteq f(x) \sqcap f(y)$.*

Definição 2 *Uma Estrutura Monotônica de Análise de Fluxo de Dados consiste de uma tripla (L, F, s_inicio):*

(1) L é um semi-reticulado de comprimento finito, com operação "meet" $\sqcap$, elemento mínimo $\perp$ e ordem parcial $\sqsubseteq$;

(2) $s\_inicio \in \{\perp, \top^1\}$; s_inicio representa a asserção sobre o programa, inicialmente válida na entrada do nó s;

(3) F é um conjunto de funções monotônicas, totais, de L para L e fechadas sobre a composição; além disto é requerido que para todo elemento $x \in L$ exista uma função de F tal que $f(s\_inicio) = x$.

Definição 3 *Uma instância $I(G, M)$ de um problema de análise de fluxo de dados consiste de:*

(1) uma estrutura monotônica de análise de fluxo de dados (L, F, s_inicio);

(2) um grafo de fluxo de controle $G(N, E, s)$, tal que $\forall n \in N$ existe pelo menos um caminho a partir de s até n;

(3) um mapeamento $M : E \to F$, onde M associa uma função de F a cada arco $e \in E$; denota-se $M(e)$ por f_e.

Estende-se a notação para incluir f_P, a função associada com o caminho P, da seguinte maneira: se P é um caminho vazio então $f_P(x) = x$; se $P = (n_1, n_2, \ldots, n_k, n_{k+1})$, então $f_P(x) = f_{e_k} f_{e_{k-1}} \ldots f_{e_1}(x)$ onde $e_1 = (n_1, n_2), \ldots, e_k = (n_k, n_{k+1})$.

A solução desejada para os problemas de fluxo de dados é a atribuição "meet over all paths" (MOP) que informalmente significa a máxima informação atribuída a cada nó ao longo de todo possível caminho de execução a partir do nó início até o nó em questão. Formalmente, a atribuição MOP pode ser definida como o mapeamento $MOP : N \to L$ tal que para todos os nós $n \in N$, $MOP(n) = \sqcap\{f_P(s\_inicio) \mid P$ *é um caminho de* s *para* $n\}$.

Kam e Ullman [KAM77] provam que, em geral, a atribuição MOP é indecidível. Entretanto, para algumas classes de *estruturas*, a solução encontrada pelos algoritmos contidos em [HEC77], [KAM77] e [HOR87], é a atribuição MOP. Os problemas modelados através de *estruturas* distributivas têm a solução MOP na saída dos algoritmos acima.

Definição 4 *Uma estrutura (L, F, s_inicio) é chamada de distributiva sse para qualquer f em F temos:*

$$\forall x, y \in L, f(x \sqcap y) = f(x) \sqcap f(y).$$

[1] Elemento máximo do semi-reticulado L.

4 Modelamento da Determinação dos Potenciais Du-Caminhos como uma Estrutura Monotônica de Análise de Fluxo de Dados

Atualmente, está sendo definida uma ferramenta de teste para apoio à utilização dos critérios Potenciais Usos. Uma tarefa que se impõe, para o apoio à seleção e avaliação de conjuntos de casos de teste, é a determinação dos caminhos requeridos por estes critérios. Neste sentido, é fundamental que se determinem os potenciais du-caminhos entre o nó s, onde ocorre a definição de um conjunto de variáveis representado por $defg(s)$, e um nó j qualquer pertencente a N, ou seja, a determinação do grafo(s). O grafo(s) é um grafo $G(N', E', s)$, onde $N' \subseteq N$ e $E' \subseteq E$, que fornece todos os potenciais du-caminhos c.r.a. qualquer variável definida em s para todo nó e todo arco alcançável a partir de s [MAL88a], [MAL88b]. O fato de programa que iremos atribuir a cada nó j do grafo de fluxo de controle é o conjunto dos potenciais du-caminhos entre o nó início s e o particular nó j. Observe-se que, para cada nó i de N tal que $defg(i) \neq \phi$, ter-se-á um novo problema de determinação dos potenciais du-caminhos, ou seja, a determinação do grafo(i), onde i é o nó início de um grafo $G'(N',E',s')$ tal que $N' \subseteq N$, $E' \subseteq E$ e $s' = i$.

O problema de determinação dos potenciais du-caminhos pode ser formalizado como uma estrutura monotônica distributiva de análise de fluxo de dados. A seguir, propõe-se uma tripla $POT - DU - CAMINHOS = (L, F, s\_inicio)$ com o objetivo desta formalização.

Seja um grafo def GD representado por $G(N, E, s)$ tal que a partir de s existe pelo menos um caminho de s até qualquer $j \in N$; associado a cada $j \in N$ está o seu conjunto $defg(j)$.

Define-se um semi-reticulado L com operação "meet" $\sqcap$, como se segue:

(1) $L \subset 2^{CAMINHOS}$, onde $CAMINHOS$ = conjunto de todos os caminhos do grafo G a partir do nó início s.

$L = 2^{Cs}$, onde $Cs = \{(s, n_1, \ldots, n_m, j) \mid (s, n_1, \ldots, n_m)$ é um caminho livre de laços e $m \geq 0$ e $deff(s, n_1, \ldots, n_m, j) \neq \phi$ e $s =$ nó início $\}$

(2) $\sqcap$ é a operação de união de conjuntos

(3) $\sqsubseteq$ é a relação entre $p, q \in L$ tal que $p \sqsubseteq q$ sse $q \subseteq p$.

Definição 5 *Definem-se as funções componentes $f : L \rightarrow L$ do conjunto F, da estrutura POT-DU-CAMINHOS da seguinte maneira:*

(1) $e_e(x) = x$, $\forall x \in L \wedge \forall e \in E$;

(2) $z_e(x) = \phi$, $\forall x \in L \wedge \forall e \in E$;

(3) $g_e(x)$ é tal que $\forall x \in L \wedge e = (n_m, j) \in E$:

$$g_e(x) = \begin{cases} \{(s, n_1, \ldots, n_m, j) \mid (s, n_1, \ldots, n_m) \text{ é um caminho livre de laço} \in x \text{ onde } m \geq 1 \wedge deff(s, n_1, \ldots, n_m, j) \neq \phi\}^2 \\ \phi \quad \text{se } \neg\exists(s, n_1, \ldots, n_m) \text{ livre de laço} \in x \vee (\exists(s, n_1, \ldots, n_m) \text{ livre de laço} \in x \wedge deff(s, n_1, \ldots, n_m, j) = \phi)^3 \end{cases}$$

(4) $h_e(x) = x \cup \{(s, n_1)\}$ *onde* $e = (s, n_1) \in E$;
(5) $\forall u, v \in F$ *então* $l(x) = u(x) \cup v(x)$; *e*
(6) $\forall q, t \in F$, $m(x) = q(t(x)) \in F$.

A seguir prova-se agora que a tripla (L, F, s_inicio) proposta acima é uma estrutura monotônica distributiva de análise de fluxo de dados.

Lema 1 *Sejam L um semi-reticulado e* $f_1, f_2, \ldots, f_n$ *funções em L. Se é verdade que para* $(\forall x, y \in L)(\forall 1 \leq i \leq n)[f_i(x \sqcap y) \sqsubseteq f_i(x) \sqcap f_i(y)]$, *então* $f_1 f_2 \ldots f_n(x \sqcap y) \sqsubseteq f_1 f_2 \ldots f_n(x) \sqcap f_1 f_2 \ldots f_n(y)$.

Prova: encontra-se em [KAM77].•

Lema 2 *Sejam L um semi-reticulado e* $f_1, f_2, \ldots, f_n$ *funções em L. Se é verdade que para* $(\forall x, y \in L)(\forall 1 \leq i \leq n)[f_i(x \sqcap y) = f_i(x) \sqcap f_i(y)]$, *então* $f_1 f_2 \ldots f_n(x \sqcap y) = f_1 f_2 \ldots f_n(x) \sqcap f_1 f_2 \ldots f_n(y)$.

Prova: análoga à do Lema 1.•

Lema 3 *Seja* $l(x) = v(x) \cup u(x)$ *onde* $u, v \in F$. *Se* $u(x \sqcap y) = u(x) \sqcap u(y)$ *e* $v(x \sqcap y) = v(x) \sqcap v(y)$ *então* $l(x \sqcap y) = l(x) \sqcap l(y)$.

Prova: $l(x \sqcap y) = v(x \sqcap y) \cup u(x \sqcap y) = (v(x) \sqcap v(y)) \cup (u(x) \sqcap u(y)) = (v(x) \cup v(y)) \cup (u(x) \cup u(y)) = v(x) \cup u(x) \cup v(y) \cup u(y) = l(x) \cup l(y) = l(x) \sqcap l(y)$.•

Teorema 1 *A tripla* $POT - DU - CAMINHOS = (L, F, s\_inicio)$ *é uma estrutura monotônica distributiva de análise de fluxo de dados.*

Prova: É trivial verificar que $(L, \sqcap)$, onde $\sqcap$ é a operação de união de conjuntos, é um semi-reticulado de comprimento finito com ordem parcial $\sqsubseteq$ e cujos elementos mínimo e máximo são Cs e ϕ, respectivamente. Assim, a condição (1) da Definição 2 é satisfeita.

[2] Observe-se que $g_e(x)$ vai levar a um conjunto de potenciais du-caminhos derivado dos caminhos livres de laço que chegam até n_m acrescentando-se o nó j, desde que existam variáveis, definidas em s, "vivas" em n_m, nestes caminhos.

[3] Ou seja, $g_e(x)$ vai levar ao conjunto ϕ se não existirem caminhos livre de laço que cheguem até n_m; ou, se existirem tais caminhos, não existem variáveis definidas em s "vivas" em n_m, nestes caminhos.

A asserção sobre o programa, inicialmente verdadeira na entrada do nó início s, é a inexistência de potenciais du-caminhos de s para s, ou seja, $s\_inicio = \phi$ (elemento máximo do semi-reticulado). Portanto, $s\_inicio \in \{\bot, \top\}$ e a condição (2) da Definição 2 é satisfeita.

Para verificar a condição (3) da definição 2, inicialmente mostra-se que todas as funções f em F são monotônicas. Para que todas as funções de F sejam monotônicas basta que $e_e(x), z_e(x), g_e(x), h_e(x)$ e $l(x)$ o sejam; pois, pelo Lema 1, tem-se que $\forall q, t \in F$, $m(x) = qt(x)$ é monotônica se $q(x)$ e $t(x)$ forem monotônicas. Sejam $x, y \in L$, então:

(3.1) $e_e(x \sqcap y) = x \sqcap y = x \cup y = e_e(x) \cup e_e(y) = e_e(x) \sqcap e_e(y)$. Portanto, $e_e(x \sqcap y) = e_e(x) \sqcap e_e(y)$, o que implica que $e(x \sqcap y) \sqsubseteq e_e(x) \sqcap e_e(y)$;

(3.2) $z_e(x \sqcap y) = \phi = \phi \cup \phi = z_e(x) \cup z_e(y) = z_e(x) \sqcap z_e(y)$. Portanto, $z_e(x \sqcap y) = z_e(x) \sqcap z_e(y)$, o que implica que $z_e(x \sqcap y) \sqsubseteq z_e(x) \sqcap z_e(y)$;

(3.3) $g_e(x \sqcap y) = g_e(x \cup y)$

$$g_e(x \cup y) = \begin{cases} \{(s, n_1, \ldots, n_m, j) \mid (s, n_1, \ldots, n_m) \text{ é livre de laço} \in (x \cup y) \text{ onde } m \geq 1 \wedge deff(s, n_1, \ldots, n_m, j) \neq \phi \} \\ \phi \quad \text{se } \neg\exists(s, n_1, \ldots, n_m) \text{ livre de laço} \in (x \cup y) \vee (\exists(s, n_1, \ldots, n_m) \text{ livre de laço} \in (x \cup y) \wedge deff(s, n_1, \ldots, n_m, j) = \phi); \end{cases}$$

$g_e(x) \sqcap g_e(y) = g_e(x) \cup g_e(y)$

$$g_e(x) \cup g_e(y) = \begin{cases} \{(s, n_1, \ldots, n_m, j) \mid ((s, n_1, \ldots, n_m) \text{ é um caminho livre de laço} \in x \text{ onde } m \geq 1 \wedge deff(s, n_1, \ldots, n_m, j) \neq \phi) \vee ((s, n_1, \ldots, n_m) \text{ é livre de laço} \in y \text{ onde } m \geq 1 \wedge deff(s, n_1, \ldots, n_m, j) \neq \phi)\} \\ \phi \quad \text{se } (\neg\exists(s, n_1, \ldots, n_m) \text{ livre de laço} \in x) \vee (\exists(s, n_1, \ldots, n_m) \in x \wedge deff(s, n_1, \ldots, n_m, j) = \phi)) \wedge (\neg\exists(s, n_1, \ldots, n_m) \text{ livre de laço} \in y) \vee (\exists(s, n_1, \ldots, n_m) \in y \wedge deff(s, n_1, \ldots, n_m, j) = \phi)); \end{cases}$$

Inspecionando $g_e(x \sqcap y)$ e $g_e(x) \sqcap g_e(y)$ vê-se facilmente que $g_e(x \sqcap y) = g_e(x) \sqcap g_e(y)$. Portanto, $g_e(x \sqcap y) \sqsubseteq g_e(x) \sqcap g_e(y)$.

(3.4) $h_e(x \sqcap y) = (x \sqcap y) \cup \{(s, n_1)\} = x \cup y \cup \{(s, n_1)\} = (x \cup \{(s, n_1)\}) \cup (y \cup \{(s, n_1)\}) = h_e(x) \cup h_e(y) = h_e(x) \sqcap h_e(y)$. Portanto, $h_e(x \sqcap y) \sqsubseteq h_e(x) \sqcap h_e(y)$.

(3.5) Seja F_P um conjunto composto das funções (1), (2), (3) e (4) propostas acima, $F_{PC} = F_P \cup \{qt(x) \mid q, t \in F_P\}$ e $F_1 = F_{PC} \cup \{l_1(x) = u(x) \cup v(x) \mid u, v \in F_{PC}\} \cup \{qt(x) \mid q, t \in F_{PC}\}$. Generalizando para k, tem-se $F_k = F_{k-1} \cup \{l_{k-1}(x) = u(x) \cup v(x) \mid u, v \in F_{k-1}\} \cup \{qt(x) \mid q, t \in F_{k-1}\}$ onde $k \geq 2$.

Observando-se que $F = F_k$ mostra-se abaixo, de forma indutiva, que as funções $l_k(x) \in F$ são monotônicas.

Hipótese de Indução: $l_{k-1}(x) \in F_{k-1}$ são monotônicas.

Base: De (3.1), (3.2), (3.3), (3.4) e do Lema 1 tem-se que as funções de F_{PC} são monotônicas. Observa-se ainda que $\forall l_1(x) \in F_1$, $l_1(x) = u(x) \cup v(x)$ onde $u, v \in F_{PC}$. Portanto, do Lema 3, $\forall l_1(x) \in F_1$ é monotônica. Adicionalmente, conclui-se que $\forall f \in F_1$ é monotônica.

Passo de Indução: Por hipótese, l_{k-1} é monotônica o que implica que F_{k-1} contém somente funções monotônicas. Assim, do Lema 1 e do Lema 3 tem-se que $l_k(x) \in F_k$ é monotônica.

Dessa maneira, conclui-se que $l(x)$ é monotônica.

O conjunto F de funções é fechado em relação a composição de funções, por construção, restando-nos mostrar que para qualquer $x \in L$ temos que existe $f \in F$ tal que $f(s\_inicio) = x$.

Para qualquer $P = (s, n_1, \ldots, n_m, j)$ onde $P \in Cs$ e $e_1 = (s, n_1) \ldots$ e $e_k = (n_m, j)$, é fácil ver que $f_P(\phi) = g_{e_k} g_{e_{k-1}} \ldots h_{e_1}(\phi) = \{P\}$. Então $\forall x \in L$ tal que $x = \{P, P', \ldots, P^n\}$ existem $f_P(\phi) = \{P\}$, $f_{P'}(\phi) = \{P'\}, \ldots$ e $f_P^n(\phi) = \{P^n\}$ onde $f_P, f_{P'}, \ldots, f_{P^n} \in F$. A função $l^n(\phi) = x$ é obtida da seguinte maneira:

$$\exists l'(x) \in F \mid l'(\phi) = f_{P'}(\phi) \cup f_P(\phi)$$

$$\exists l''(x) \in F \mid l''(\phi) = f_{P''}(\phi) \cup l'(\phi)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\exists l^n(x) \in F \mid l^n(\phi) = f_{P^n}(\phi) \cup l^{n-1}(\phi) = x$$

Assim, verificou-se a condição (3) da definição 2.

Para mostrar que $POT - DU - CAMINHOS$ é distributiva, basta observar que $e_e(x), z_e(x), g_e(x), h_e(x)$ e $l(x)$ são distributivas (sub-itens (3.1), (3.2), (3.3), (3.4), (3.5) e Lema 3) e que m(x) também o é pelo Lema 2. Portanto, conclui-se que $POT - DU - CAMINHOS = (L, F, s\_inicio)$ é uma estrutura monotônica distributiva de análise de fluxo de dados.•

Teorema 2 *Considere-se uma instância I(G',M) da estrutura distributiva POT-DU-CAMINHOS definida da seguinte maneira:*

Seja um grafo def obtido a partir de um programa Q definido pelo grafo de fluxo de controle G(N,E,s) e os conjuntos defg(i) associados a todo $i \in N$. Define-se $G'(N', E', s')$[4] tal que $N' = N$ e $E' = E \cup \{(s, s)\}$. O mapeamento $M : E' \to F$ é definido a seguir:

Seja $e = (i, j) \in E'$,

$$M(e) = \begin{cases} \mathbf{2}_e(x) & , se\ e = (s, s) \in E'; \\ h_e(x) & , se\ e = (s, n_1) \in E'; \\ g_e(x) & , se\ e = (n_m, j) \in E' \wedge n_m \neq s. \end{cases}$$

A solução MOP para a instância I(G',M) acima atribui para cada nó i de N' todos os potenciais-du-caminhos de s até i, portanto, o grafo(s) é obtido de $\cup\{MOP(i) \mid i \in N\}$.

[4] Essa modificação no grafo G gerando o grafo G' é feita para garantir a existência de pelo menos um caminho de e para s.

Prova: a solução MOP é definida pelo mapeamento $MOP : N' \rightarrow L$ tal que $\forall i \in N'$, $MOP(i) = \sqcap\{f_P(s\_inicio) \mid P$ *é um caminho de s para i* $\}$. Se o caminho $P = (s, n_1, \ldots, n_m, i) \in Cs$ então é fácil ver que $f_P(s\_inicio) = f_{e_k} f_{e_{k-1}} \cdots f_{e_1}(s\_inicio) = \{P\}$ dado o mapeamento M proposto e $sinicio = \phi$. Semelhantemente, se $P \in CAMINHOS - Cs$ então $f_P(\phi) = f_{e_k} f_{e_{k-1}} \cdots f_{e_1}(\phi) = \phi$. Como a operação "meet" $\sqcap$ é definida como a união de conjuntos e os caminhos P pertencentes a Cs são potenciais du-caminhos temos que $MOP(i) =$ { *todos potenciais du-caminhos de s para i* } .•

4.1 Custo da Determinação dos Potenciais Du-caminhos

Seja um programa Q escrito na linguagem de implementação de Rapps e Weyuker [RAP85]. O grafo def GD obtido a partir desse programa é representado por $G(N, E, s)$ e os conjuntos $defg(i)$, $i \in N$. Para cada $i \in N$ tal que $defg(i) \neq \phi$, temos um grafo def GD_i representado por um grafo de fluxo $G_i(N_i, E_i, s_i)$ tal que $N_i \subseteq N$, $E_i \subseteq E$ e $s_i = i$ e os conjuntos $defg(k)$, $k \in N_i$; onde N_i é o conjunto de nós pertencentes a N e que são alcançáveis a partir de i. Dessa maneira, $\forall j$, $j \neq i$, existe pelo menos um caminho de i para j e $\forall (j,k) \in E_i$, $j, k \in N_i$. Note-se que o grafo(i) definido na Seção 2 é um sub-grafo do grafo $G_i(N_i, E_i, s_i)$ de GD_i.

Para determinar o potenciais du-caminhos do programa Q deve-se aplicar o algoritmo que calcula a solução MOP para cada instância I_i construída a partir de GD_i. Logo, o custo de determinação dos potenciais du-caminhos de Q é a soma do custo da aplicação do algoritmo para cada instância I_i.

Para determinar a solução MOP para cada instância $I_i(G_i', M_i)$ é necessário que o algoritmo percorra todos caminhos $(i, n_1, \ldots, n_m, j)$, $j \in N'$, tal que $(i, n_1, \ldots, n_m)$ é livre de laço, pois caminhos com laços não habilitam a formação de novos elementos do semi-reticulado L . Dessa maneira, a complexidade do algoritmo será proporcional ao número de potenciais du-caminhos que começam por i.

Portanto, o custo de determinação do potenciais du-caminhos *(CUSTO)* é dado por:

$$CUSTO = \sum_{\forall i | defg(i) \neq \phi} \textit{Custo do Algoritmo para a instância } I_i.$$

Porém, o custo do algoritmo para a instância I_i é O(número de potenciais du-caminhos que começam por i). Logo, podemos concluir que o custo de determinação dos potenciais du-caminhos é O(Soma de todos potenciais du-caminhos do programa Q).

Maldonado, Chaim e Jino mostram em [MAL88b] que, alocando estrategicamente as definições de variáveis de um programa escrito na linguagem proposta por Rapps e Weyuker [RAP85], o grafo de fluxo de controle apresentado na Fig. 1 leva ao maior número de potenciais du-caminhos. Este número foi calculado e é igual a $((11/2)t + 9)2^t - 10t - 9$, onde t é o número de transferências condicionais.

Assim, tem-se que o custo de determinação dos dos potenciais du-caminhos é proporcional a $t2^t$; pois, no pior caso, ter-se-á que a soma de todos os potenciais du-caminhos é o número dado acima.

5 Conclusões

Mostrou-se neste trabalho que o problema de determinação dos potenciais du-caminhos, requeridos pelos critérios Potenciais Usos, é um problema de análise de fluxo de dados e se enquadra dentro das estruturas monotônicas de análise de fluxo de dados. Adicionalmente, mostrou-se que a solução MOP resolve o problema de determinação dos potenciais du-caminhos. Estes resultados permitem que se utilize os algoritmos de Horwitz, Demers e Teitelbaum [HOR87] para a obtenção da solução MOP e consequentemente a resolução deste problema.

Ainda os resultados anteriores permitem fazer considerações sobre o custo, para o pior caso, da determinação dos potenciais du-caminhos. Este custo é proporcional a $t2^t$ onde t é o número de transferências condicionais do programa. Cabe ressaltar, entretanto, que este é o valor para o pior caso; para programas comuns este valor deve ficar abaixo do valor de pior caso dado por $((11/2)t + 9)2^t - 10t - 9$.

6 Agradecimentos

Este trabalho foi parcialmente suportado por CNPq, CAPES e SID Informática S.A..

7 Referências

[FOS76] - L. D. Fosdick and L. J. Osterweil, "Data Flow Analysis in Software Reliability," *ACM Computing Surveys*, vol. 8, pp. 305-330, 1976.

[FRA88] - F. G. Frankl and E.J. Weyuker, "An Applicable Family of Data Flow Testing Criteria," *IEEE Trans. Software. Eng.*, Vol. 14, No.10, Oct. 1988, pp. 1483-1498.

[HEC77] - M. S. Hecht, *Flow Analysis of Computer Programs*, North Holland: Amsterdam, 1977.

[HER76] - P. M. Herman "Data Flow Approach to Program Testing," *Australian Computer Journal*, vol. 8, No, 3, Nov. 1976.

[HOR87] - S. Horwitz, A. Demers and T. Teitelbaum, "An Efficient General Iterative Algorithm for Dataflow Analysis," *Acta Informatica*, vol. 24, pp. 679-694, 1987.

[KAM77] - J. B. Kam and J. D. Ullman, "Monotone Data Flow Frameworks," *Acta Informatica*, vol. 7, pp. 305-317, 1977.

[LAS83] - J. W. Laski e B. Korel, "A Data Flow Oriented Program Testing Strategy," *IEEE Trans. Software. Eng.*, Vol. SE - 9, No.3, Maio 1983, pp. 347-354.

[MAL88a] - J. C. Maldonado, M. L. Chaim, M. Jino, "Seleção de Casos de Teste baseada nos Critérios Potenciais Usos," in *Proc. II Simp. Bras. de Eng. de Software*, Canela, R.S., Out.,1988, pp. 24-35.

[MAL88b] - J. C. Maldonado, M. L. Chaim, M. Jino, "Resultados do Estudo de uma Família de Critérios de Teste de Programas baseado em Fluxo de Dados," Relatório Técnico Interno RT/DCA/001/88 - DCA/FEE/Unicamp - Dez. 1988.

[MYE79] - G. Myers, *The Art of Software Testing*, Wiley: New York, 1979.

[NTA84] - S. C. Ntafos, "On Required Element Testing," *IEEE Trans. Software Eng.*, Vol. SE - 10, pp. 795-803, Nov. 1984.

[RAP85] - S. Rapps and E. J. Weyuker, "Selecting Software Test Data Using Data Flow Information," *IEEE Trans. Software Eng.*, vol. SE - 11, No. 4, pp. 367-375, Abril, 1985.

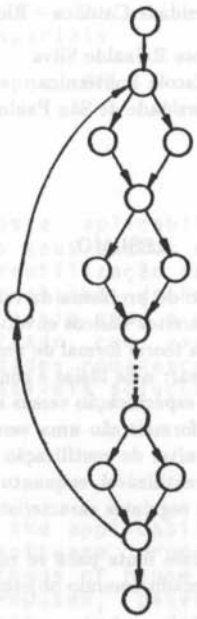

Figura - 1 *Grafo de Fluxo de Controle que maximiza o Número de Potenciais Ducaminhos.*