D8-1

Preorder Closures for Nondeterministic Programs

徐 良為[†] 武市 正人[††] 岩崎 英哉[†††]
Liangwei XU Masato TAKEICHI Hideya IWASAKI

†東京大学大学院 工学系研究科 情報工学専攻
Information Engineering Course, Graduate School, Division of Engineering, University of Tokyo
††東京大学 工学部計数工学科
Dept. Mathematical Engineering, University of Tokyo
†††東京大学 教育用計算機センター
Educational Computer Centre, the University of Tokyo

概 要

In this paper, we give a relational description of nondeterministic programs in so-called "Unbounded Nondeterministic Iterative Programming Language". Some algebraic properties about nondeterministic iterative loops, and relationships between preorder closures and loops are explored. This paper also gives an approach of designing nondeterministic programs on relations.

1 Introduction

Nondeterminism has been recognized as a vital component of designing programs for a variety of computation architectures [5].

Relation, endowed with nondeterministic nature, is suitable for describing nondeterministic program languages. In addition, relation serves a specification of a program when it is regarded as a logic formula between input and output values of the program. Therefore, programming based on a relational specification is reduced to refining one relation (specification) into another (it's implementation). A relation R is said to be a refinement of another relation S when R preserves the total correctness of S, i.e., intuitively, when R and S are fed with the same input value, all possible results of R must also be included in the results of S. In addition, the definition domain of R should not be smaller than S.

In this paper, we give a relational description of a nondeterministic programming language and show how to refine a specification relation into a relation which is composed only by these operators permitted by the target programing language.

For example, in defining a specification, one may use an angelic composition to combine two relations R and S $(R \bullet S)$ where $S = \{(2,1),(3,1),(4,2)\}$ and $R = \{(1,2),(2,4)\}$. The result of $R \bullet S$ is $\{(1,1),(2,2)\}$. From the implementation point of view, such a composition may need a backtrack when R is passed 3 from S to fail its computation (S is given an input 1). One way to solve this problem is to refine such the angelic composition into the one what is called demonic composition which need no backtracking.

This paper is organized as follows. After introducing some preliminary definitoins and exploring some important properties about the demonic operators, we present a relational definition of a nondeterministic programming language. Then, we illustrate our approach of relational programming through a simple example — The Parallel

Linear Search. Finally, some concluding remarks are presented.

2 The Calculus of Relations

A *binary relation* is a subset of a cartesian product. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary sets. A relation R having the type "$\mathcal{A}$ from $\mathcal{B}$" or $(\mathcal{A} \leftarrow \mathcal{B})$ satisfies $R \subseteq \mathcal{A} \times \mathcal{B}$. We write aRb as a shorthand for $(a,b) \in R$. The *angelic composition* of two relations $R \in \mathcal{A} \leftarrow \mathcal{B}$ and $S \in \mathcal{B} \leftarrow \mathcal{C}$ is the relation $(R \bullet S) \in \mathcal{A} \leftarrow \mathcal{C}$ which is defined as
$\forall a \in \mathcal{A},\ c \in \mathcal{C} :\ a(R \bullet S)c \equiv \exists b \in \mathcal{B} :\ aRb \wedge bSc$.
There are two special relations in $\mathcal{A} \leftarrow \mathcal{B}$: the *empty relation* $\emptyset = \{\}$ and the *top relation* $\top = \mathcal{A} \times \mathcal{B}$. The *complement* of a relation R is defined as $\tilde{R} = \{\ (y,x) \mid \neg\ (yRx)\ \}$. The *converse* of a relation R is defined as $R^\circ = \{\ (y,x) \mid xRy\ \}$.

The *product* of two relations is defined as $(y_1,y_2)(R \times S)(x_1,x_2) \equiv y_1Rx_1 \wedge y_2Sx_2$ and $(y,z) < R,S > x \equiv yRx \wedge zSx$. There are two projections π_1 and π_2 which are defined as $\pi_1.(a,b) = a$ and $\pi_2.(a,b) = b$.

We also denote the set of all these relations with the same source type and target type as $\mathcal{R}_\mathcal{A} = \{\ R \mid R \subseteq \mathcal{A} \times \mathcal{A}\ \}$. There is an *identity* relation $I = \{\ (x,x) \mid x \in \mathcal{A}\ \}$ in $\mathcal{R}_\mathcal{A}$.

A relation P is called a *monotype* if $P \subseteq I$. There is a one to one correspondence between a predicate p over $\mathcal{A}$ and a monotype $P = \{\ (x,x) \mid p.x\ \}$ in $\mathcal{R}_\mathcal{A}$. The complement of a monotype P is also a monotype $\overline{P}$ which is defined as $\overline{P} = \tilde{P} \cap I$. The *domain* and the *codomain* of a relation R are defined as $R\Box = \{\ (x,x) \mid \exists y :\ yRx\ \}$ and $\Box R = \{\ (y,y) \mid \exists x :\ yRx\ \}$ respectively. Obviously, both the domain and the codomain of a relation are monotypes.

A relation R is said to be a *function* if it is *entire* $(I \subseteq R^\circ \bullet R)$ and *simple* $(R \bullet R^\circ \subseteq I)$. Because a function f is simple, the application of f to an argument x is uniquely defined and is written as $f.x$.

2.1 Recursion and Fixed points

A function f from relations to relations is said to be *monotonic* if it respects relational inclusion:

$$\forall R,S :\ \ S \subseteq R \ \Rightarrow\ f(S) \subseteq f(R)\ .$$

According to the result of Tarski [9], the formula $f.X \subseteq X$ over a complete lattice has at least one solution when f is monotonic. We denote the least solution of the formula as $(\mu X :\ f.X)$. The meaning of the "least" is as follows.

$$\forall Y :\ \ f.Y \subseteq Y \ \ \Rightarrow\ \ \mu X : f.X \ \subseteq\ Y\ .$$

2.2 Preorder Closures

A *preorder* is a relation S that is both reflexive $(I \subseteq S)$ and transitive $(S \bullet S \subseteq S)$. For any relation $R \in \mathcal{A} \leftarrow \mathcal{A}$, there exists a smallest preorder R^* containing R. We call it the *preorder closure* of R. The preorder closure can also be defined as (Backhouse [3]):

$$R^* = (\mu X :\ I \ \cup\ X \bullet R).$$

We also have

$$S \bullet R^* \ =\ \mu X :\ (S \ \cup\ X \bullet R).$$

As a correspondence to the demonic closure (in the following section), we also denote $\mu X :\ (S \ \cup\ X \bullet R)$ as $S * R$.

2.3 Demonic Composition and Demonic Closure

In defining a nondeterministic system, we need a "non-standard" relational operator: *demonic composition* $(R \odot S)$. The demonic composition is defined as

$$R \odot S \ =\ R \bullet S \bullet (R \triangleright S),$$

where $R \triangleright S$ is a monotype which is the greatest solution of $X\ :\ S \bullet X \subseteq R\Box \bullet S$. We call the operator $\triangleright$ as a *demonic limitation*.

As a correspondence to the preorder closure, we define the *demonic closure* of two relations S and R as follows.

$$S \circledast R \ =\ \mu X :\ S \ \cup\ X \odot S$$

Since the function $X \mapsto S \ \cup\ X \odot S$ is monotonic, according to the Tarski theorem, we know that the least fixed point of the function do exist.

(4). Loop

$$[\![\textbf{Do } S \textbf{ oD}]\!] = \overline{[\![S]\!]\Box} \circledast [\![S]\!]$$

(5). Parallel Composition

$$[\![(\textbf{Do } S_1 \textbf{ oD}) \;||\; (\textbf{Do } S_2 \textbf{ oD})]\!] = \overline{([\![S_1]\!] \cup [\![S_2]\!])\Box} \circledast ([\![S_1]\!] \cup [\![S_2]\!])$$

The predicate P in the guarded statement is interpreted as a monotype, i.e., $[\![P]\!] = \{(x,x) | P.x\} \subseteq I$.

5 Parallel Linear Search

Given a predicate f over integers, we are to design a program to find a point at which the f yields *true*. The example was prompted by Apt [8] and Knapp [7] in which they used two different program transformation methods to demonstrate the correctness of a solution to the problem. We show that by using relational calculus.

5.1 Formal Definition

We denote the specification relation as Pls (Parallel linear search). From the above definition, we know that Pls can formally be defined as follows.

$$\forall x, y \in Int: \quad y \; Pls \; x \equiv f.y$$

As we know, for any $x \in Int$, y satisfies $y \geq x \vee y \leq x$. Hence, Pls can also be defined as

$$y \; Pls \; x \equiv (y \geq x \wedge f.y) \quad \vee \quad (y \leq x \wedge f.y)$$

It is known that the relation $(\geq)$ and $(\leq)$ over integers are equivalent to the preorders $(+1)^*$ and the $(-1)^*$ in respect. We conclude that Pls can also be defined in a relational form:

$$Pls = f? \bullet (+1)^* \cup f? \bullet (-1)^*$$

where $f?$ is the monotype corresponding to the predicate f.

5.2 Derivation

We refine Pls as follows.

Pls

$=$ { Def. }

$f? \bullet (+1)^* \cup f? \bullet (-1)^*$

$=$ { Tupling (the theorem (**T10**)) }
{ Def. $S = f? \bullet \pi_1 \cup f? \bullet \pi_2$ }
{ Def. $R = (+1) \times I \cup I \times (-1)$ }
{ $< I, I >$ is a function }

$(S \bullet R^*) \odot < I, I >$

$\geq$ { the theorem (**T3**) }

$S \odot (\overline{(R \bullet \overline{S\Box})\Box} * (R \bullet \overline{S\Box})) \odot < I, I >$

$=$ { the theorem (**T9**) }

$S \odot (\overline{(R \bullet \overline{S\Box})\Box} \circledast (R \bullet \overline{S\Box})) \odot < I, I >$

The result can be rewritten as a nondeterministic program as follows ($\overline{S\Box} = \overline{f?} \times \overline{f?}$).

$$Pls = \textbf{Begin } < I, I >; \; Loop; \; S \; \textbf{End}$$
$$Loop = \textbf{Do } q \rightarrow ((+1) \times I \;[\!]\; I \times (-1)) \textbf{ oD}$$
$$q.(a,b) = \neg (f.a) \wedge \neg (f.b)$$
$$S = ((\pi_1; f) \rightarrow \pi_1) \;[\!]\; ((\pi_2; f) \rightarrow \pi_2)$$

6 Concluding Remarks

We have shown a method of using relations to describe nondeterministic programs. Our program notions are similar to UNITY [5] or Action System [2]. In our notion, the fairness assumption of a loop is implicitly expressed by the demonic closure — a recursively definition based on demonic compositions and angelic choices. The approach permits us to design nondeterministic programs in a common base — relational calculus.

参考文献

[1] R.J.R. Back and R. Kurki-Suonio: *Decentralization of process nets with centralized control.* In 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, pp. 131-142, ACM, 1983.

[2] R.J.R. Back: *Refinement Calculus, Part I and II: Parallel and Reactive Programs.* In Lecture Notes in Computer Science val. 527.

[3] R. Backhouse and P. Hoogendijk: *Elements of Relational Theory of Datatypes.* In Lecture Notes in Computer Science val. 755. (1993).

[4] R. Backhouse: *Demonic Operators and Monotype Factors.* Mathematical Structures in Computer Science, 3(4):417-433, December 1993.

[5] K.M. Chandy and J.Misra: *Parallel Program Design: A Foundation.* Addison-Wesley, Reading, MA, 1988.

[6] E.W. Dijkstra and C.S. Scholten: *Predicate Calculus and Program Semantics.* Springer-Verlag, Berlin, 1990.

[7] E. Knapp: *Derivation of concurrent programs: two examples.* Science of Computer Programming 19(1992). pp.1-23.

[8] E.R. Olderog and K.R. Apt: *Using Transformation to Verify Parallel Programs.* in: Algebraic Methods II: Theory, Tools and Applications, LNCS 490 (Springer, Berlin, 1991) pp. 55-81.

[9] A. Tarski: *On the calculus of relations.* Journal of Symbolic Logic, 6(3):73-89,1941.

2.4 Refinement Orders

Finally, we introduce two refinement orders. A relation R is said to be a refinement of relation S if one of the following conditions is satisfied:

(1). $R\Box = S\Box \ \wedge \ R \subseteq S$

(2). $R\Box \supseteq S\Box \ \wedge \ R \bullet S\Box \subseteq S$

We denote $R \leq S$ if they satisfy (1) and $R \preceq S$ if they satisfy (2).

The definition of $\preceq$ is essentially equivalent to the definition of refinement relation in Back [1] which is based on the Dijkstra's weakest precondition. Both $\leq$ and $\preceq$ preserve the total correctness. The $\preceq$ is monotonic with respect to the demonic composition, whereas the $\leq$ is monotonic with respect to the union.

3 Some Algebraic Properties about Demonic Closure

In this section, we explore some properties of the demonic closure. As we have mentioned that the demonic operators are not the "standard" operators in the relational calculus, they, , unlike the angelic operators, have little algebraic properties.

In the following theorems, let $R, S, T \in \mathcal{R}_{\mathcal{A}}$ be arbitrary relations and P be an arbitrary monotype.

The following theorem (**T1**) shows an approximation about the demonic loop.

T1 $\forall n \geq 0 : \ (\overline{R\Box} \circledast R)\Box \supseteq (\overline{R\Box} \odot R^n)\Box$
where R^n *is inductively defined as* $R^0 = I$; $R^{n+1} = R \bullet R^n$.

The theorem (**T2**) shows that the $\circledast$ is left monotonic.

T2 $R \circledast T \subseteq S \circledast T \ \Leftarrow \ R \subseteq S$

The theorem (**T3**) and theorem (**T4**) show some refinement orders between closures.

T3 $S * (R \bullet \overline{S\Box}) \leq S * R$

T4 $\overline{R\Box} \circledast R \preceq \overline{S\Box} \circledast S \ \Leftarrow \ R \leq S$

The following theorems show relationships between demonic closures and the preorder closures.

T5 $S \circledast R \subseteq S * R, \qquad I \circledast R = R^*$

T6
$(S * R) \bullet (S \circledast R)\Box = S \circledast R$
$\Leftarrow \ R\Box \bullet S\Box = \emptyset$

T7 $T \circledast R = T * R \ \Leftarrow \ R$ *is simple.*

T8 $S \circledast R = S * R \ \Leftarrow \ S\Box \supseteq \Box R$

T9 $\overline{R\Box} \circledast R = \overline{R\Box} * R \ \Leftarrow \ R$ *is well-founded*

The following theorem is useful in combining two loops into one.

T10 *(Tupling)*
$R_1 \bullet R^* \cup S_1 \bullet S^*$
$= T \bullet (R \times I \cup I \times S)^* \bullet < I, I >$
where $T = R_1 \bullet \pi_1 \cup S_1 \bullet \pi_2$

4 A Nondeterministic Programming Language

In this section, we define a small nondeterministic programming language by relational calculus. The language is similar to UNITY (Chandy [5]) or Action System (Back [2]), except that we use functions instead of assignment statement.

4.1 Syntax

Prog	::=	**Begin** S	**End**
S	::=	*Functions*	(functions)
	|	S ; S	(demonic comp.)
	|	$S \parallel S$	(angelic choice)
	|	$Pred \rightarrow S$	(guarded stat.)
	|	*Loop*	
	|	*Loop* || *Loop*	(loop union)
Loop	::=	**Do** S **oD**	

4.2 Semantics

The semantics of a statement S is denoted by a relation $[\![S]\!] \in \mathcal{R}_{\mathcal{A}}$.

(1). Demonic Composition

$$[\![S_1; S_2]\!] = [\![S_2]\!] \odot [\![S_1]\!]$$

(2). Angelic Choice

$$[\![S_1 \parallel S_2]\!] = [\![S_1]\!] \cup [\![S_2]\!]$$

(3). Guarded Statement

$$[\![P \rightarrow S]\!] = [\![S]\!] \bullet [\![P]\!]$$