ソフトウェア工学 116－1
（1997. 9. 11）

Many-sorted Propositional Dynamic Logic for Parallel Processing Environment

Wu Guoqing Tetsuo Tamai
Graduate School of Arts and Sciences, University of Tokyo, Tokyo , Japan 153
E-mail: wu@graco.c.u-tokyo.ac.jp E-mail: tamai@graco.c.u-tokyo.ac.jp

Abstract This paper presents a many-sorted propositional dynamic logic system (MPDL) describing dynamic properties of programs in parallel processing environment, and applies MPDL to describe a practical example with respect to a distributed system. Differing from normal PDL, a new operator G^{ij} is used in the paper to describe relation between parallel programs. Finally, the paper simply discusses consistency and completeness of MPDL.

keyword Many -sorted propositional dynamic logic, Configuration, Parallel processing environment, Cooperation point.

1. Introduction

An important characteristic of the parallel processing environments ("PPE" for short) is that parallel executing programs not only can independently solve problems but also may jointly solve a complex problem by means of cooperation. The characteristic gives rise to a new research topic for researchers who studied propositional dynamic logic("standard PDL" for short) . In the past, standard PDL studied dynamic properties of programs was confined to a objective fact , that is, there is only one processor in the computer system and the processor only can execute a action(instruction) of a program in many concurrent executing programs (or processes) at a time. But , now many parallel executing programs in PPE can execute different acts simultaneously at a time on separate processors. Hence, it is obvious that PDL for studying dynamic properties of programs in PPE must be different from standard PDL. For the this reason, this paper is to be presented a many-sorted propositional dynamic logic system("MPDL" for short) for PPE, and applies it to describe a practical example with respect to a distributed system.

2. Formal Parallel Processes Environment

For the convenience of description, we assume that there are k processors and may simultaneously execute k programs in a PPE, and call ith parallel executing program in this PPS ith kind of program.

We now define the syntax of MPDL. Because standard PDL can be used to describe dynamic properties of programs in ith processor, symbols we use in here are similar to [1] and [2], and express a^i, b^i, ... for ith kind of atomic programs, A^i, B^i, ... for ith kind of atomic formulas or symbols of propositional variables. Two underlying sets are defined as follows:

$AP^i = \{ \phi, a^i, b^i, \cdots \}$; ($\phi$ is idle program), $AF^i = \{ \text{true}, \text{false}, A^i, B^i, \cdots \}$.

We inductively define set of formulas, For^i, and set of programs, $Prog^i$, by the following rules, where α^i, β^i,... and X^i, Y^i, ... are compound programs and formulas, respectively.

Porgrams: (1) $AP^i \subseteq \text{Prog}^i$;

(2) if α^i, $\beta^i \in \text{Prog}^i$, then $\alpha^i\beta^i$, $\alpha^i \cup \beta^i$, $(\alpha^i)^* \in \text{Prog}^i$, where

$$(\alpha^i)^* = (\alpha^i)^0 \cup (\alpha^i)^1 \cup L = \bigcup_{j=0}^{\infty}(\alpha^i)^j.$$

Formulas: (1) $AF^i \subseteq For^i$;

(2) if X^i, $Y^i \in For^i$ and $\alpha^i \in Prog^i$, then $\neg X^i$, $X^i \vee Y^i$, $<\alpha^i> X^i \in For^i$.

Kripke structure M^i of ith kind of standard PDL is a triple (S^i, $|=^i$, $\xrightarrow[i]{}$), where

$$|=^i:\quad AF^i \to 2^{S^i}; \qquad \xrightarrow[i]{}:\ AP^i \to 2^{S^i \times S^i}.$$

Informally, S^i is a set of ith kind of program states. The function $|=^i$ provides an interpretation for ith kind of atomic formulas, that is, $t^i \in |=^i (A^i)$ means "A^i is true at the state t^i or t^i is satisfied with A^i in M^i" and it may be expressed as $M^i, t^i |=^i A^i$. The function $\xrightarrow[i]{}$ provides an interpretation for the ith kind of atomic programs, that is, $(s^i, t^i) \in \xrightarrow[i]{}(a^i)$ means " there is an execution of a^i which begins in state s^i and ends in state t^i" and it may be expressed as $s^i \xrightarrow[i]{a^i} t^i$.

Note that the idle program Φ does not result in a state change, that is, $(t^i, t^i) \in \xrightarrow[i]{}(\Phi)$. Furthermore, symbol i in notion " $\xrightarrow[i]{}$ " may be omitted where it dose not result in misunderstanding .

According to the definition of ith kind of PDL, a program state $\vec{t}$ in PPE which is constituted by k processors is made up of k kinds of program states, that is, $\vec{t} = (t^1, \ldots, t^k)$ and $t^i \in S^i$. We also call $\vec{t}$ as a state configuration of PPE ("configuration" for short). By the definition of configuration an executing sequence of PPE can be defined as follows:

Definition 1. An executing sequence T of PPE is either an infinite or a finite sequence of configurations, T = $\vec{t}_0$, $\vec{t}_1$, ... , where $\vec{t}_0 = (t_0^1, \cdots, t_0^k)$ is the starting configuration and $\vec{t}_{j+1}$ is the next configuration of $\vec{t}_j$. Let $a = (a^1, a^2, \ldots, a^k)$ and $\vec{t}_j \xrightarrow{a} \vec{t}_{j+1}$ is an executing step in PPE, the executing step of ith kind of program is $t_j^i \xrightarrow{a^i} t_{j+1}^i$. It is obvious that at most k atomic programs can execute simultaneously in each executing step.

Definition 2. For PPE which has k parallel executing programs , Kripke structure M corresponding to it is a triple (S,|=,→), where S is a set of configurations and S = $S^1 \times S^2 \times \ldots \times S^k$. Let Q_k= {1,2,..., k}, $\vec{t}_1, \vec{t}_2 \in$ S and $\vec{t}_1 = (t_1^1, \cdots, t_1^k)$, $\vec{t}_2 = (t_2^1, \cdots, t_2^k)$, X is a formula of MPDL(defined in next section) and X^i is ith kind of formula appearing in X. Symbol " |= " means that $\vec{t}_1$ |= X in M if and only if $\exists i \in Q_k$ ($M^i, t_1^i |=^i X^i$) . Symbol "→ " means that $\vec{t}_1 \xrightarrow{a} \vec{t}_2$ if and only if $\forall i \in Q_k$($t_1^i \xrightarrow{a^i} t_2^i$).

To reduce difficulties in resolving complex problems and improve efficiency , some parallel executing programs in PPE are required to cooperate. So these programs shall communicate each other or share resources at some point of time . We call this point of time cooperative point. To describe this cooperative point, a new relation R^{ij} is used in structure M, that is, for any $t^i \in S^i$ and $t^j \in S^j$, $t^i R^{ij} t^j$ means that there is a cooperation between ith program at position t^i and jth program at position t^j in a configurations $\vec{t}$ = ($t^1, \ldots, t^i, \ldots, t^j, \ldots, t^k$) . For example, ith program at position t^i makes a request to jth program at position t^j for cooperation, or jth program at position t^j reports cooperative result to ith program at position t^i. Hence structure M has to be expanded as follows:

$$M=(S, \models, \rightarrow, \{R^{ij} \mid i,j \in Q_k\})$$

where $S, \models, \rightarrow$, are defined as before. For R^{ij}, $\forall i,j \in Q_k(R^{ij} \subseteq S^i \times S^j)$ and R^{ij} has following properties: Let $i,j,h \in Q_k$, $t^i \in S^i$, $t^j \in S^j$, $\vec{t} = (t^1, \cdots, t^i, \cdots, t^j, \cdots, t^k)$,

(1) $R^{ij} \cdot R^{jh} \subseteq R^{ih}$;

(2) when $p = t^x$ or $p = (t')^j$, if $(t^i R^{ij} p)$ then $p = t^j$;

(3) $t^i R^{ii} t^x$ if and only if ("iff" for short) $\exists j \neq i\,(t^j = t^x \wedge t^i R^{ij} t^j)$.

According to relation R^{ij}, new operator G^{ij} is defined as follows:

Definition 3. For any $i,j \in Q_k$, if $X \in For^j$, then $G^{ij}X \in For^i$ and

$$M,\vec{t} \models G^{ij}X \text{ iff } M^i,t^i \models G^{ij}X \text{ iff } \exists t^j(t^i\ R^{ij}\ t^j \wedge M^j,t^j \models X)$$

Furthermore, we define that $W^{ij} \equiv \neg G^{ij} \neg X$.

3. Axiom System of MPDL

Definition 4. Let Prog and For are sets of programs and formulas of MPDL, respectively. Prog and For are defined inductively by the following rules:

prog: (1) $\forall i \in Q_k (Prog^i \subseteq Prog)$,

(2) if $\alpha^1,\cdots,\alpha^k \in Prog$, *then* $\alpha = (\alpha^1,\cdots,\alpha^k) \in Prog$,

(3) if $\alpha,\beta \in$ Prog, then $\alpha\beta = (\alpha^1\beta^1,\cdots,\alpha^k\beta^k) \in$ Prog, $\alpha \cup \beta = (\alpha^1 \cup \beta^1,\cdots,\alpha^k \cup \beta^k) \in$ Prog, $(\alpha)^* = ((\alpha^1)^*, (\alpha^2)^*,\cdots, (\alpha^k)^*) \in Prog$.

For: (1) $\forall i \in Q_k (For^i \subseteq For)$,

(2) if $X_1, X_2 \in For$ and $\alpha \in Prog$, then $X_1 \vee X_2, \neg X, <\alpha> X \in$ For,

(3) $i,j \in Q_k$, if $X \in For^j$ and $G^{ij}X \in For^i$, then $G^{ij}X \in$ For.

Now, we use the structure M to interpret semantics of Prog and For:

Let $\vec{t},\vec{q} \in S$, $\alpha,\beta \in$ Prog, $\alpha^i,\beta^i \in Prog^i$, $X^i \in For^i$, $i \in Q_k$, $X,Y \in For$,

(1) $\vec{q} \xrightarrow{\alpha} \vec{t}$ iff $\forall i(q^i \xrightarrow{\alpha^i} t^i)$; (sometimes M or M^i appearing ahead of "|=" will be omited)

(2) $\vec{q} \xrightarrow{\alpha\beta} \vec{t}$ *iff* $\exists \vec{r} \in S(\vec{q} \xrightarrow{\alpha} \vec{r}$ *and* $\vec{r} \xrightarrow{\beta} \vec{t})$ *iff* $\exists \vec{r} \in S \forall i(q^i \xrightarrow{\alpha^i} r^i$ *and* $r^i \xrightarrow{\beta^i} t^i)$

(3) $\vec{q} \xrightarrow{\alpha\cup\beta} \vec{t}$ *iff* $\vec{q} \xrightarrow{\alpha} \vec{t}$ *or* $\vec{q} \xrightarrow{\beta} \vec{t}$ *iff* $\forall i(q^i \xrightarrow{\alpha^i} t^i$ *or* $q^i \xrightarrow{\beta^i} t^i)$;

(4) $\vec{q} \xrightarrow{\alpha^*} \vec{t}$ iff $\exists \vec{r}_1,\cdots,\vec{r}_n \in S(\vec{q} = \vec{r}_1 \xrightarrow{\alpha} \vec{r}_2 \xrightarrow{\alpha} \cdots \xrightarrow{\alpha} \vec{r}_n = \vec{t})$

iff $\exists \vec{r}_1,\cdots,\vec{r}_n \in S \forall i(q^i = r_1^i \xrightarrow{\alpha^i} r_2^i \xrightarrow{\alpha^i} \cdots \xrightarrow{\alpha^i} r_n^i = t^i)$;

(5) $M,\vec{t} \models X^i$ iff $\exists t^i \in S^i(M^i, t^i \models X^i)$;

(6) $\vec{t} \models X \vee Y$ iff $\vec{t} \models X$ or $\vec{t} \models Y$; (7) $\vec{t} \models \neg X$ iff $\vec{t} \not\models X$;

(8) $\vec{t} \models <\alpha> X$ iff $\exists \vec{q}(\vec{t} \xrightarrow{\alpha} \vec{q}$ *and* $\vec{q} \models X)$; (9) $\vec{t} \models G^{ij} X$ (same to definition 3).

we also write $[\alpha]X \equiv \neg\langle\alpha\rangle\neg X$ and $[\alpha]X \in For$.

Definition 5. A formula X is true (or valid) in M, that is, $M \models X$, iff $\forall \vec{t} \in S(M,\vec{t} \models X)$.

Based on previous definitions, described semantics and Segerberg axiom system, the axiom system of MPDL can be defined as follows: Let kinds of α, β, X,Y have been determined and $i,j \in Q_k$,

(1) for any i all ith kind of propositional tautologic; (2) $<\alpha\cup\beta>X \equiv <\alpha>X \vee <\beta>X$;
(3) $<\alpha>$false $\equiv$ false; (4) $<\alpha>(X\vee Y) \equiv <\alpha>X \vee <\alpha>X$;
(5) $<\alpha\beta>X \equiv <\alpha><\beta>X$; (6) $<\alpha^*>X \equiv X\vee<\alpha><\alpha^*>X$;
(7) $<\alpha^*>X \supset X\vee<\alpha^*>(\neg X \wedge <\alpha>X)$; (8) $G^{ij}X \supset W^{ij}X$;
(9) $G^{ij}X \vee G^{ij}Y \equiv G^{ij}(X\vee Y)$; (10) $G^{ij}G^{jh}X \supset G^{ih}X$;
(11) $G^{ij}\text{true} \supset \bigvee_{j\neq i} G^{ij}\text{true}$; (12) $G^{ij}X \supset X$.

The rules of inference are:

(1) $\dfrac{X, X\supset Y}{Y}$, (2) $\dfrac{X}{[\alpha]X}$, (3) $\dfrac{X}{W^{ij}X}$ $(j\neq i)$.

For axioms (1) ~ (7), discussions of validity completely are the same as standard PDL besides kinds of programs and formulas are considered. For example, let X∈For and X^i is ith kind of formula in X and i∈Q_k for axiom (3):

$$\vec{t} \models <\alpha\cup\beta>X \Leftrightarrow \exists\vec{q}(\vec{t} \xrightarrow{\alpha\cup\beta} \vec{q} \wedge \vec{q}\models X) \Leftrightarrow \exists\vec{q}\forall i(t^i \xrightarrow{\alpha^i\cup\beta^i} q^i \wedge q^i \models X^i)$$
$$\Leftrightarrow \exists\vec{q}\forall i(((t^i \xrightarrow{\alpha^i} q^i)\wedge q^i \models X^i)\vee((t^i \xrightarrow{\beta^i} q^i)\wedge q^i \models X^i)) \Leftrightarrow \vec{t} \models <\alpha>X\vee<\beta>X.$$

Because of limited space, we will only discuss those axioms with new operator G^{ij}. For axioms(8)~(12):

$$\vec{t}\models G^{ij}X \Rightarrow t^i\models G^{ij}X \Rightarrow \exists t^j(t^iR^{ij}t^j \wedge t^j\models X) \Rightarrow \neg\exists t^j(t^iR^{ij}t^j \wedge t^j\models \neg X) \Rightarrow t^i\models \neg G^{ij}\neg X \Rightarrow \vec{t}\models W^{ij}X.$$
$$\vec{t}\models G^{ij}X\vee G^{ij}Y \Leftrightarrow \exists t^j(t^iR^{ij}t^j \wedge t^j\models X) \vee \exists t^j(t^iR^{ij}t^j \wedge t^j\models Y) \Leftrightarrow \exists t^j(t^iR^{ij}t^j \wedge t^j\models X\vee Y) \Leftrightarrow \vec{t}\models G^{ij}(X\vee Y).$$
$$\vec{t}\models G^{ij}G^{jh}X \Rightarrow \exists t^j(t^iR^{ij}t^j \wedge t^j\models G^{jh}X) \Rightarrow \exists t^j(t^iR^{ij}t^j \wedge \exists t^h(t^jR^{jh}t^h \wedge t^h\models X))$$
$$\Rightarrow \exists t^j\exists t^h(t^iR^{ij}t^j \wedge t^jR^{jh}t^h \wedge t^h\models X) \Rightarrow \exists t^h(t^iR^{ih}t^h \wedge t^h\models X) \Rightarrow \vec{t}\models G^{ih}X.$$
$$\vec{t}\models G^{ij}\text{true} \Rightarrow \exists t^{i'}(t^iR^{ij}t^{i'} \wedge t^{i'}\models true) \Rightarrow \exists t^{i'}\exists j\neq i(t^j = t^{i'} \wedge t^iR^{ij}t^j \wedge t^j\models \text{true})$$
$$\Rightarrow \bigvee_{j\neq i}(t^iR^{ij}t^j \wedge t^j\models \text{true}) \Rightarrow \vec{t}\models \bigvee_{j\neq i} G^{ij}true.$$
$$\vec{t}\models G^{ij}X \Rightarrow \exists t^j(t^iR^{ij}t^j \wedge t^j\models X) \Rightarrow \exists t^j(t^j\models X) \Rightarrow \vec{t}\models X.$$

Lemma 1. Assume X is a formula of MPDL, A_1, ..., A_m are atomic formulas in X and have n kinds(n≦ $|Q_k|$). if M |= X , then there are at most n M^{i_j}'s (i_j≦n, j≦m) such that $M^{i_j} \models A_j$.

Proof. By definition 5, M |= X iff $\forall\vec{t}\in S(M,\vec{t}\models X)$. Furthermore, by $\vec{t}=(t^1,\cdots,t^k)$, there certainly exists a i_j so that $M, t^{i_j}\models A_j$ (A_j is a atomic formula in X and i_jth kind). By $t^{i_j}\in S^{i_j}$, then $M^{i_j}, t^{i_j}\models A_j$. Because configuration $\vec{t}\in S$ is unrestricted, $M^{i_j}\models A_j$. When kinds of atomic formulas are n, by i_j≦n, thus there are at most n M^{i_j}'s corresponding to kinds. ■

Theorem 1. Rules (1)~(3) of inference guard truth of formulas.

Proof. For the rule(1), assume that X and Y are ith kind of formulas, X⊃Y is true and Y is false in M, that is, for any state $\vec{t}\in S$, $M,\vec{t}\models X$, $M,\vec{t}\models X\supset Y$, $M,\vec{t}\models\neg Y$. By lemma1, $M^i,t^i\models X$, $M^i,t^i\models X\supset Y$ and $M^i,t^i\not\models Y$. Furthermore, if $M^i,t^i\models X$, then $M^i,t^i \not\models \neg X$ so that $M^i,t^i \not\models X\supset Y$. This result is in contradiction with preconditions. Hence, $M,\vec{t}\models Y$. When X and Y are many-kinds of formulas (that is, X and Y are made up of many kinds of atomic formulas), we also can similarly prove rule(1) by lemma1.

For the rule(2), assume that X is a ith kind of formula and true in M, that is, for any $t^i \in S^i$, $M^i, t^i \models X$ so that $M^i, t^i \not\models \neg X \Rightarrow$ for $\alpha \in Prog^i$, $M^i, t^i \not\models <\alpha>\neg X$. $\Rightarrow M^i, t^i \models \neg<\alpha>\neg X$, that is, $M^i, t^i \models [\alpha]X$. Let X is a many-kinds of formula and $A_1, \ldots, A_n$ are atomic formulas in X, by lemma1, there certainly exists some structures M^i corresponding with kinds of atomic formulas. For a configuration $\vec{t} \in S$ if $M, \vec{t} \models X$, then $M^i, t^i \models A_j$(A_j is ith kind of formula and $j \leqq n$). So $M^i, t^i \not\models \neg A_j$. Furthermore, for $\alpha \in Prog^i$, $M^i, t^i \not\models <\alpha>\neg X$. Thus, $M, \vec{t} \not\models <\alpha>\neg X$ so that $M, \vec{t} \models [\alpha]X$.

For the rule(3), let X is a jth kind of formula and true in M, that is,

$$\forall \vec{t} \in S(M, \vec{t} \models X) \Rightarrow \exists t^j (M^j, t^j \models X) \Rightarrow \neg\exists t^j (M^j, t^j \models \neg X) \Rightarrow \neg\exists t^j (t^i R^{ij} t^j \wedge M^j, t^j \models \neg X)$$
$$\Rightarrow M^i, t^i \models \neg G^{ij}\neg X, \text{ that is }, M, \vec{t} \models \neg G^{ij}\neg X. \text{ By } W^{ij}X \equiv \neg G^{ij}\neg X,\ M, \vec{t} \models W^{ij}X. \quad \blacksquare$$

4. A Practical Example

In this section, we will discuss a practical example with respect to a distributed system using MPDL.

Example. Four programs are allowed to execute simultaneously in a distributed system. These programs can communicate each other by writing (or sending) or reading (or getting) mails and jointly solve a complex problem. This system provides a additional memory (called CRAM) to be shared by the programs. The CRAM is separated into many sections (that is , mail-box) in fixed-length. Each program write mails in or read mails from CRAM through the bus. The exclusive form is used in administration of the bus, that is, the bus is occupied only by a program at a point of time. Assigned method of the bus is implemented by hardware and ensured that the bus satisfies following fair condition:

If a program P wants to occupy the bus to write mails in (or read mails from) CRAM and CRAM is not full (or not empty), then mails that P wants to write (or read) finally can be written in (or read from) CRAM.

Now, we show some descriptions with respect to the bus:

Let OREC($F_1, \ldots, F_n$) means that $F_1, \ldots, F_n$ are exclusive and Q_k= { 1,2,3,4}. For the bus, we assume bus is a program administering CRAM and Cth kind. The set of propositional variables with respect to the bus is AF^c = { Idle, { $ReqR^i$, $ReqW^j$ | i, j $\in Q_k$ }, Write, Read, Empty, Full }. The members of AF^C mean the bus is idle, ith or jth kind of program made a request to the bus for reading or writing mails, the bus is writing or reading mails, CRAM is empty or full, respectively.

For each kind of programs, they all relate to the bus when they want to read or write mails through the bus. Furthermore, for any i $\in Q_k$, all requests of ith kind of programs that want to write (or read) mail in (or from) CRAM must pass through the atomic program a^i_S (or a^i_G) to the bus. Thus, we also assume that $G^{ic}ReqW^i$ and $G^{ic}ReqR^i$ are included in propositional set of ith kind of program. Note that

$G^{ic}ReqW^i$ and $G^{ic}ReqR^i$ means the state that ith kind of program has send a request to the bus for write (or read) mails and is waiting for respond of the bus.

By the description above, formal expressions with respect to the bus can be described as follows:

For the bus: 1) OREC(Idle, Write, Read); 2) OREC(Idle, { G^{iC} true | $i \in Q_k$});

3) $ReqW^i \supset$ Write, $ReqR^i \supset$ Read; 4) Write $\vee$ Read $\supset \neg$Idle.

For the administration of CRAM, we only discuss two cases that CRAM is full or empty :

5) Write $\supset \neg$Empty, Read $\supset \neg Full$; 6) $\neg$(Full $\wedge$ Empty);

7) $\bigvee_{i \in Q_k} (ReqW^i) \wedge Empty \supset$ Write; 8) $\bigvee_{i \in Q_k} (ReqR^i) \wedge Full \supset$ Read;

9) $\bigvee_{i \in Q_k} (ReqW^i) \wedge \neg Full \supset$ Write; 10) $\bigvee_{i \in Q_k} (ReqR^i) \wedge \neg Empty \supset$ Read.

For the fair condition:

11) $(< a_S^i > G^{iC} ReqW^i \supset \text{false}) \supset$ Full; 12) $(< a_G^i > G^{iC} ReqR^i \supset \text{false}) \supset$ Empty

According to MPDL and formal description above, we discuss the activity problem of this system.

Proposition 1. $D \equiv \text{Idle} \supset ((\bigwedge_{i \in Q_k} (\neg ReqW^i)) \wedge (\bigwedge_{j \in Q_k} (\neg ReqR^j))) \vee ((\bigwedge_{i \in Q_k} (\neg ReqW^i)) \wedge Empty)$

$\vee ((\bigwedge_{j \in Q_k} (\neg ReqR^j)) \wedge Full)$ is provable.

D means: if the bus is idle, then either no any program request to write or read mails, or CRAM is empty(full) but no any program request to write(read) mails.

Proof.

(1) $(\bigvee_{i \in Q_k} ReqW^i) \wedge \neg Full \supset Write$ (by 9)) (2) $(\bigvee_{i \in Q_k} ReqR^j) \wedge \neg Empty \supset Read$ (by 10))

(3) $\neg \text{Write} \supset \neg((\bigvee_{i \in Q_k} ReqW^i) \wedge \neg Full)$ (by (1)) (4) $\neg \text{Write} \supset \bigwedge_{i \in Q_k} (\neg ReqW^i) \vee Full$

(5) $\neg \text{Read} \supset \neg((\bigvee_{j \in Q_k} ReqR^j) \wedge \neg Empty)$ (by (2)) (6) $\neg \text{Read} \supset \bigwedge_{j \in Q_k} (\neg ReqR^j) \vee Empty$

(7) $\text{Write} \vee \text{Read} \supset \neg \text{Idle}$ (by 4)) (8) $\text{Idle} \supset \neg \text{Write} \wedge \neg \text{Read}$

(9) $\text{Idle} \supset (\bigwedge_{i \in Q_k} (\neg ReqW^i) \vee Full) \wedge (\bigwedge_{j \in Q_k} (\neg ReqR^j) \vee Empty)$

(10) $\text{Idle} \supset ((\bigwedge_{i \in Q_k} (\neg ReqW^i)) \wedge (\bigwedge_{j \in Q_k} (\neg ReqR^j))) \vee ((\bigwedge_{i \in Q_k} (\neg ReqW^i)) \wedge Empty) \vee (\bigwedge_{j \in Q_k} (\neg ReqR^j)) \wedge Full)$. ■

Also, because of limited space, we will not attempt to discuss descriptions and proofs of other properties with respect to this example system.

5. The Consistency and Completeness of MPDL

Let X be a many-kinds formula of MPDL and M is Kripke structure of MPDL.

Definition 6. MPDL is consistent, if there is not a formula X in MPDL such that both $\vdash_{MPDL} X$ and $\vdash_{MPDL} \neg X$, where $\vdash_{MPDL} X$ and $\vdash_{MPDL} \neg X$ means that X and $\neg$X are provable in MPDL, respectively. Below, "MPDL" in symbol " $\vdash_{MPDL}$ " will be omitted.

Theorem 2. For any formula X, if $\vdash X$, then $M \models X$

Proof. Because X is a many-kinds of formula, all axioms of MPDL is true in M and rules (1) ~ (3) of inference can guard truth of formulas (by theorem 1), so when $\vdash X$ is given, we can prove $M \models X$ using induction hypothesis for complexity of X. ■

Theorem 3. MPDL is consistent.

Proof. Assume MPDL is not consistent, then there are $\vdash X$ and $\vdash \neg X$ in MPDL simultaneously. By theorem 2, there are also both $M,\vec{t} \models X$ *and* $M,\vec{t} \models \neg X$. When X is a single kind or many-kinds of formula, there is certainly $i \in Q_k$ so that $M^i, t^i \models a^i$ and $M^i, t^i \models \neg a^i$ (a^i is an ith kind of atomic formula appeared in X). Thus a^i and $\neg a^i$ all hold at state t^i simultaneously. A contradiction is produced. Similarly, other kinds of atomic formulas appeared in X also can be proved. Hence, it is impossible that there are $\vdash X$ and $\vdash \neg X$ in MPDL simultaneously. We can obtain the result that MPDL is consistent. ■

For the completeness of MPDL, we will use Kozen's proof method[1]. Because completeness of MPDL is not focal point of this paper, we will only principally discuss some reviews with respect to MPDL.

Definition 7. A formula X is consistent in MPDL, if X is provable and not $\vdash \neg X$ in MPDL.

Definition 8. Let W is a many-kinds of formula and consistent, $\alpha, \beta \in$ Prog, FL(W) is the smallest set of formulas containing W so that:

(1) all of kinds of atomic formulas appeared in $W \in FL(W)$;

(2) if $X \vee Y \in FL(W)$, then $X, Y \in FL(W)$;

(3) if $\neg X, \langle \alpha \rangle X \in FL(W)$, then $X \in FL(W)$;

(4) if $\langle \alpha \cup \beta \rangle X \in FL(W)$, then $\langle \alpha \rangle X, \langle \beta \rangle X \in FL(W)$;

(5) if $\langle \alpha^* \rangle X \in FL(W)$, then $X, \langle \alpha \rangle\langle \alpha^* \rangle X \in FL(W)$;

(6) if $\langle \alpha \beta \rangle X \in FL(W)$, then $\langle \alpha \rangle\langle \beta \rangle X \in FL(W)$;

(7) if $G^{ij}X \in FL(W)$, then $X \in FL(W)$.

It is easy to see that FL(W) is finite.

Definition 9. Let $FL(W) = \{X_1, \ldots, X_n\}$ and the configuration $\vec{t}$ of FL(W) is $Y_1 \wedge Y_2 \wedge \ldots \wedge Y_n$, where each Y_h ($h \leqq n$) is either X_h or $\neg X_h$. For the convenience of following proof and consistent with contents described as before, we will use symbols $\vec{t}, \vec{q}, \vec{r}, \ldots$ and $t^i, t^j, \ldots$ denote configurations of FL(W) and conjunction of ith, jth, ... kind of formulas appeared in configuration $\vec{t}$ such that $\vec{t} = t^i \wedge t^j \wedge \ldots$, respectively.

For any $X \in FL(W)$ and $\vec{t}$, it is obvious that $\vdash \vec{t} \supset X$ *or* $\vdash \vec{t} \supset \neg X$ because either X or $\neg X$ appears in $\vec{t}$. We also write $\vec{t} \leq X$ if $\vdash \vec{t} \supset X$.

Definition 10. Kripke structure of FL(W) is $M' = (S', \models, \rightarrow)$, where S' is a set of configurations of FL(W) (similar to S in M), meanings of symbols "$\models$" and "$\rightarrow$" are similar to proceeding definitions. For any kind of atomic formula A^i, $\vec{t} \models A^i$ *iff* $\vec{t} \leq A^i$, and for any atomic program $a = (a^1, a^2, \ldots, a^k)$, $\vec{t} \xrightarrow{a} \vec{q}$ iff

$\vec{t} \wedge < a > \vec{q}$ is consistent.

Lemma 2. For any program α, If $\vec{t} \wedge < \alpha > \vec{q}$ is consistent, then $\vec{t} \xrightarrow{\alpha} \vec{q}$.

Lemma 3. For any $<\alpha>X \in FL(W)$ and configuration $\vec{t}$, $\vec{t} \leq <\alpha> X$ iff $\exists \vec{q}((\vec{t} \xrightarrow{\alpha} \vec{q}) \wedge \vec{q} \leq X)$.

Lemma 4. Let Y is a jth kind of formula. For any $G^{ij}Y \in FL(W)$ and $\vec{t}$,

$$\vec{t} \leq G^{ij}Y \text{ iff } \exists t^j(t^i R^{ij} t^j \wedge t^j \leq Y).$$

Lemma 5. For any formula $X \in FL(W)$ and configuration $\vec{t}$, $\vec{t} \models X$ iff $\vec{t} \leq X$.

For proofs of lemma 2 – lemma 5, because axioms, definitions, rules of inference and constituted rules of FL(W) defined as before , and induction hypothesis chiefly are used , we will omit details of proofs .

Now, we can easily prove the following theorem using the definitions and theorems above.

Theorem 4. MPDL is complete.

Proof. Since W is a many-kinds of formula of MPDL and consistent, By lemma 5, the completeness of MPDL can be proved. ■

6. Conclusion

Based on the standard PDL, this paper discussed MPDL which formally defined parallel processing environment , and also applied MPDL to describe a simple practical example . Since our main research objective is how to expand the standard PDL to MPDL so that we can study dynamic properties of programs in parallel processing environment with MPDL, this research topic is how to expand functions of PDL. We believe that contents of the paper will be helpful to formal researches with respect to parallel processing environment . Also, we will further deal with other dynamic properties of programs in parallel processing environment.

References

[1] Dexter Kozen, Rohit Parikh, An elementary proof of the completeness of PDL, Theoretical Computer Science, 14(1981): pp113–118.

[2] Fischer M.J. and Ladher R.E., Propositional dynamic logic for regular program, J. Computer system ScienceVol.18, No.2, 1979: pp194–211.

[3] Entalbert P., Many–Sorted Temporal logic for Multi–Processes System, LNCS, 176, Berlin, Springer–verlag,1984.

[4] K. Abrahamson, Modal Logic of Concurrent Nondeterministic Programs, LNCS, 70, 1979.

[5] Z. Manna and P.Wolper, Synthesis of Communicating Process from Temporal Logic Specification, ACM Tran. on Programming Language and System, Vol.6, No.1,1984.