TR-511

An Operational Semantics of And-Or-Parallel Logic Programming Language, ANDOR-II

by
A. Takeuchi & K. Takahashi (Mitsubishi)

October, 1989

An Operational Semantics of And- Or-Parallel Logic Programming Language, *ANDOR-II* [*]

Akikazu Takeuchi, Kazuko Takahashi
Mitsubishi Electric Corp.
8-1-1, Tsukaguchi honmachi
Amagasaki, Hyogo 661
Japan

September 12, 1989

Abstract

An operational semantics of and- or-parallel logic programming language, ANDOR-II, is presented. ANDOR-II combines or-parallel computation of Prolog and and-parallel computation of committed choice logic programming languages such as PARLOG, Concurrent Prolog and Guarded Horn Clauses.

Starting from a naive semantics suitable for simulating in sequential machines, we develop a new semantics with fine grain parallelism. The semantics is based on the coloring scheme which paints variable substitutions made in each or-parallel world by distinct colors.

1 Introduction

The target of ANDOR-II project is 1) to design a language having both and-, or-parallelism suitable for parallel problem solving and 2) to find a class of and-, or-parallelism which can be translated into committed-choice languages such as GHC[Ueda] efficiently. The second is worth investigating since it might reveal the complexity of small grain parallel and distributed implementation of and- or-parallel computation.

Design and implementation of a language ANDOR-II which meets two requirements above was already finished and reported together with its application[Takeuchi]. ANDOR-II is now implemented in such a way that an ANDOR-II program is translated into a GHC program first and then the translated program is executed by a GHC processor. In this paper an underlying operational semantics of ANDOR-II computation which realizes fine grain parallelism is presented.

The paper is organized as follows. First, a more general and- or-parallel language than ANDOR-II is given in Section 2. Section 3 and 4 provide two sequential operational semantics to that language, respectively. Section 3 presents a simple computation model which is relatively

*This research was done as part of the Fifth Generation Computer Project and supported by the Institute for New Generation Computer Technology.

well known as *branching world model* and is suitable for intuitive understanding of and- or-parallel computation. It can be characterized by the term *eager copying*. The model is used in many implementation of and-, or-parallel languages because of its simplicity. Section 4 presents a new computation model, called a *vector model*, which allows multiple bindings generated to the same variables by independent or-parallel worlds. The model is regarded as *lazy copying*. Section 5 discusses about the difference between the two models. Section 6 discusses about such language restriction that simplifies the vector model. Section 7 handles the problems encountered when parallelizing the vector model. Also the implementation in a Committed-Choice Language (CCL), especially GHC, is described.

Here is a listing of all symbols used in this paper.

Atoms	:	A, B, G
Clauses	:	C
Head of a clause C	:	$head(C)$
Body of a clause C	:	$body(C)$
Programs	:	P
Variables	:	X, Y, Z
Worlds	:	w
Sets of worlds	:	W
Colors	:	α, β
Colored atoms	:	$\mathcal{A}, \mathcal{B}, \mathcal{G}$
Unifiers	:	θ
Colored unifiers	:	Θ
Conjuctions of colored atoms	:	$\mathcal{W}$

All these symbols above may be used with superscripts and/or subscripts. The principle of super(sub)scripting is as follows:

$$\Gamma^{\{\text{disjunctive numbering}\}}_{\{\text{conjunctive numbering}\}}$$

Therefore if Γ is an atom, A^1 and A^2 are in disjunctive relation, and B_1 and B_2 are in conjunctive relation. Also θ^1 and θ^2 indicate that they are logically independent and hence making their composition is useless.

2 An and- or-parallel logic programming language

A predicate is classified into either an *and-relation* or an *or-relation*. A definition of an and-relation consists of guarded clauses. A definition of an or-relation consists of non-guarded clauses.

A guarded clause is of the form:

$$G_0 :- G_1, \ldots, G_n \mid B_1, \ldots, B_m.$$

where G_0 and $G_1, \ldots, G_n$ and $B_1, \ldots, B_m$ are called a *head*, a *guard (part)* and a *body (part)*, respectively. An atom in a guard part is restricted to simple built-in predicates such as comparison. An atom in a body part can be an and-relation or an or-relation. A non-guarded clauses is of the form:

$$G_0 : -B_1, \ldots, B_m.$$

where G_0 and $B_1, \ldots, B_m$ are called a *head* and a *body*, respectively. An atom in a body part can be an and-relation or an or-relation.

Informally speaking, a goal reduction with guarded clauses is similar to that of GHC. A goal reduction with non-guarded clauses is similar to that of or-parallel Prolog except for that a non-guarded clause which instantiates the goal in the head unification will suspend until the goal is sufficiently instantiated.

3 Branching world model/Eager copying

A conjunction of atoms is called a *world*. Every atom A such that $A = p(t_1, \ldots, t_n)$ and p is an or-relation is associated with a set $D(A)$ of clauses defining p.

3.1 Non-labelled

Definition 1 *Let W_i be a set of worlds. Assume that $w_j(\in W_i) = A_1^j, \ldots, A_a^j$ and $A_k^j (1 \leq k \leq a)$ are chosen for reduction by computation rule. Then derivation of W_{i+1} from W_i is defined as follows:*

Case 1 *A_k^j is a unification, $X = t$:*
W_{i+1} is defined as $W_{i+1} = (W_i - \{w_j\}) \cup \{w_j{}'\}$,
where $w_j{}' = (A_1^j, \ldots, A_{k-1}^j, A_{k+1}^j, \ldots, A_a^j) \cdot \{X/t\}$.

Case 2 *A_k^j is an and-relation: If the following conditions are satisfied,*

1. *$\exists G_0$:- $G_1, \ldots, G_n \mid B_1, \ldots, B_m$ such that $\exists mgu\ \theta\ A_k^j = G_0 \cdot \theta$,*
2. *$(G_1, \ldots, G_n) \cdot \theta$ has guarded refutation and generates θ',*

then W_{i+1} is defined as $W_{i+1} = (W_i - \{w_j\}) \cup \{w_j{}'\}$,
where $w_j{}' = (A_1^j, \ldots, A_{k-1}^j, B_1, \ldots, B_m, A_{k+1}^j, \ldots, A_a^j) \cdot \theta \cdot \theta'$.

Case 3 *A_k^j is an or-relation: If there exists a non-empty set of $\{C_1, \ldots, C_t\} \subseteq D(A_k^j)$ such that $\exists mgu\ \theta^l\ A_k^j = \mathrm{head}(C_l) \cdot \theta^l, l = 1, \ldots, t$, respectively,*
then W_{i+1} is defined as

$$W_{i+1} = \begin{cases} (W_i - \{w_j\}) \cup \bigcup_{l=1,\ldots,t} \{w_j^l\} & \text{if } D_s = \phi \\ \\ (W_i - \{w_j\}) \cup \bigcup_{l=0,\ldots,t} \{w_j^l\} & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} w_j^l &= (A_1^{jl}, \ldots, A_{k-1}^{jl}, body(C_l), A_{k+1}^{jl}, \ldots, A_a^{jl}) \cdot \theta^l, \ l = 1, \ldots, t, \text{respectively,} \\ &\quad A_1^{jl}, \ldots, A_{k-1}^{jl}, A_{k+1}^{jl}, \ldots, A_a^{jl} \text{ is } l\text{-th copy of } A_1^j, \ldots, A_{k-1}^j, A_{k+1}^j, \ldots, A_a^j \\ w_j^0 &= (A_1^j, \ldots, A_{k-1}^j, A_k^j|_{D_s}, A_{k+1}^j, \ldots, A_a^j) \end{aligned}$$

$A^j_k|_{D_s}$ is an atom with a new definition D_s,
$D_s = D - D_f - \{C_1, \ldots, C_t\}$
$D_f \subset D(A^j_k)$ is a set of clauses which are not unifiable with A^j_k

3.2 Labelled

Labelling each or-parallel world is considered. A label is either a distinct symbol or a concatenation of symbols, corresponding to a branch and history of branching, respectively. A label is denoted by ω.

When a world branches into several worlds, variables appearing in the original world are copied. History of variable copying is made explicit by the structure Θ, which has the form:

$$\Theta = \{X/\langle X_1|\omega_1, \ldots, X_n|\omega_n\rangle, \ldots\}$$

The form means that a variable X_i is a copy of the variable X in the world labelled by ω_i.

Definition 2 (Labelled Branching World) *Let P be a program and let (W_i, Θ_i) be a set of worlds and history of variable copying. Assume that $w_j(\in W_i) = A^j_1, \ldots, A^j_a$ and $A^j_k (1 \leq k \leq a)$ are chosen for reduction by computation rule. Then derivation of (W_{i+1}, Θ_{i+1}) from (W_i, Θ_i) is defined as follows.*

Case 1 *A^j_k is a unification, $X = t$:*
$W_{i+1} = (W_i - \{w_j\}) \cup \{w_j{}'\}, \Theta_{i+1} = \Theta_i$,
where $w_j{}' = (A^j_1, \ldots, A^j_{k-1}, A^j_{k+1}, \ldots, A^j_a) \cdot \{X/t\}$.

Case 2 *A^j_k is an and-relation: If the following conditions are satisfied,*

1. *$\exists G_0$:- $G_1, \ldots, G_n \mid B_1, \ldots, B_m$ such that $\exists mgu\ \theta\ A^j_k = G_0 \cdot \theta$,*
2. *$(G_1, \ldots, G_n) \cdot \theta$ has guarded refutation and generates θ',*

then $W_{i+1} = (W_i - \{w_j\}) \cup \{w_j{}'\}, \Theta_{i+1} = \Theta_i$,
where $w_j{}' = (A^j_1, \ldots, A^j_{k-1}, B_1, \ldots, B_m, A^j_{k+1}, \ldots, A^j_a) \cdot \theta \cdot \theta'$.

Case 3 *A^j_k is an or-relation: If there exists a non-empty set $\{C_1, \ldots, C_t\} \subseteq D(A^j_k)$ such that $\exists mgu\ \theta^l A^j_k = head(C_l) \cdot \theta^l, l = 1, \ldots, t$, respectively, then $\Theta_{i+1} = \Theta_i \cup \Theta^1$,*

$$W_{i+1} = \begin{cases} (W_i - \{w_j\}) \cup \bigcup_{l=1,\ldots,t}\{w^l_j\} & \textit{if } D_s = \phi \\ (W_i - \{w_j\}) \cup \bigcup_{l=0,\ldots,t}\{w^l_j\} & \textit{otherwise} \end{cases}$$

where

$$\begin{aligned} w^l_j &= (A^{jl}_1, \ldots, A^{jl}_{k-1}, body(C_l), A^{jl}_{k+1}, \ldots, A^{jl}_a) \cdot \theta^l,\ l = 1, \ldots, t, \textit{ respectively}, \\ & \quad A^{jl}_1, \ldots, A^{jl}_{k-1}, A^{jl}_{k+1}, \ldots, A^{jl}_a \textit{ is } l\textit{-th copy of } A^j_1, \ldots, A^j_{k-1}, A^j_{k+1}, \ldots, A^j_a \\ w^0_j &= (A^j_1, \ldots, A^j_{k-1}, A^j_k|_{D_s}, A^j_{k+1}, \ldots, A^j_a) \end{aligned}$$

[1] Canonical form of $\Theta_i \cup \Theta$ should be introduced, but not in this paper.

$$\begin{array}{rcl} & & A^j_k|_{D_s} \textit{ is an atom with a new definition } D_s, \\ & & D_s = D - D_f - \{C_1, \ldots, C_t\} \\ & & D_f \subset D(A^j_k) \textit{ is a set of clauses which are not unifiable with } A^j_k \\ \Theta & = & \{X/\langle X^1|\omega^1, \ldots, X^t|\omega^t\rangle \mid \\ & & X \textit{ appears in } A^j_1, \ldots, A^j_{k-1}, A^j_{k+1}, \ldots, A^j_a, \\ & & \textit{and } X^l \textit{ is its } l\textit{-th copy used in } A^{jl}_1, \ldots, A^{jl}_{k-1}, A^{jl}_{k+1}, \ldots, A^{jl}_a\} \\ & & \textit{where } \omega^1, \ldots, \omega^t \textit{ are new distinct symbols.} \end{array}$$

4 Vector model/Lazy copying

Branching model is eager copying scheme. Eager copying assumes that atoms belong to the same worlds can be collected easily. Under distributed execution environment, the assumption is unacceptable because atoms are distributed over multiple resources such as processors and memories.

Vector model is based on lazy copying scheme, where copy worlds are created part by part incremantally on demand. Vector model is considered to be extendable to distributed execution environment. The extension is discussed later. In this section the vector model is described under the assumption that execution is done in the sequential manner.

The model is described keeping the resemblance with the labelled branching world model. We introduce a color and a vector which are similar to a label and history of variable copying but have slightly different meanings.

4.1 Colors and vectors

Definition 3 *A primitive color is a pair (A, P) of two symbols, where P and A are called a branching point and a branching arc, respectively.*

Two primitive colors are defined to be orthogonal iff they share the same branching point, but have different branching arcs. Two primitive colors are defined to be independent iff they have different branching points.

Definition 4 *A color is defined to be a set of primitive colors, in which no element is orthogonal to other element. An empty set is regarded as a color.*

Two colors, α, β, are defined to be orthognal iff $\exists p_1 \in \alpha, p_2 \in \beta$, p_1 and p_2 are orthogonal. Orthogonality of two colors, α, β, is denoted by $\alpha \perp \beta$.

Proposition 1 *Let α and β be colors such that $\alpha \perp \beta$. Let α' be a color such that $\alpha \subseteq \alpha'$. Then $\alpha' \perp \beta$.*

In eager copying scheme, at any moment, any world has its own label. In other words, labels are identifiers of or-parallel worlds. On the contrary, in lazy copying scheme, a world is copied part by part incrementally. A color is used to identify a world, however, it is difficult to think about an independent world since at some moment two worlds may still share some part which will be copied later. In order to identify such ongoing copying, a color is attached to a term and an atom, while a label is attached to an independent world.

Definition 5 (A Vector) *A vector is a term which is defined using the n-ary function ($n \geq 0$) symbol, $\langle \ldots \rangle$, as follows.*

$$\langle a^1|\alpha^1, \ldots, a^n|\alpha^n\rangle$$

where a^i is a term and α^j's are orthogonal colors.

When a value of a variable X is considered, there are several possible values depending on which world is concerned. A vector represents such possible values with colors of its associated world.

A term is called a colored term iff it is a vector or it has at least one vector in it. A unifier is called a colored unifier iff it has at least one vector in it.

Since in our model the distinction between a term and an atom is not essential, the concept of a vector of terms can be naturally extended to a vector of atoms. Semantically a vector of atoms, $\langle A^1|\alpha^1, \ldots, A^n|\alpha^n\rangle$, can be interpreted as disjunction of atoms associated with colors. $\langle\rangle$ is a special case and is interpreted as failure.

Note that a term, an atom or a unifier not known to be colored has the possiblity to eventually become colored.

Definition 6 *Let t and t' be terms. Let α and Θ be a color and a colored unifier, respectively. A function $f(t, \alpha) = (t', \Theta)$ is defined as follows:*

Case 1 *t is not a vector and includes at least one vector v, $v = \langle a^1|\beta^1, \ldots, a^n|\beta^n\rangle$, and $\beta^{\sigma_1}, \ldots, \beta^{\sigma_m}(\{\sigma_1, \ldots, \sigma_m\} \subseteq \{1, \ldots, n\})$ are not orthogonal to α.*

$$f(t, \alpha) = (\langle t^1[v/a^{\sigma_1}]|\alpha \cup \beta^{\sigma_1}, \ldots, t^m[v/a^{\sigma_m}]|\alpha \cup \beta^{\sigma_m}\rangle, \Theta)$$

Case 2 *$t = \langle t^{1''}|\beta^1, \ldots, t^{n''}|\beta^n\rangle$ and $\beta^{\sigma_1} \ldots \beta^{\sigma_m}$ are not orthogonal to α.*

$$f(t, \alpha) = (\langle t^{\sigma_1''}|\alpha \cup \beta^{\sigma_1}, \ldots, t^{\sigma_m''}|\alpha \cup \beta^{\sigma_m}\rangle, \phi)$$

Case 3 *t is neither a vector nor include a vector.*

$$f(t, \alpha) = (\langle t|\alpha\rangle, \phi)$$

where $t^i[v/a]$ denotes the i-th copy of t with the term v replaced by a, and Θ is a colored unifier of the form

$$\Theta = \{X/\langle X^{\sigma_1}|\beta^{\sigma_1}, \ldots, X^{\sigma_m}|\beta^{\sigma_m}\rangle \mid X \text{ is any variable appearing in } t \text{ except for in } v \text{ and } X^i \text{ is the i-th copy of } X\}$$

Definition 7 *Let t and t' be terms, and let Θ be a unifier. A function $\text{flatten}(t) = (t', \Theta)$ is defined as follows:*

1. *Let F_0 be $\{(t, \phi)\}$ and S_0 be ϕ.*
2. $F_{n+1} = \bigcup_{(t''|\alpha) \in F_n} f_1(t'', \alpha)$ *and*
   $S_{n+1} = \bigcup_{(t''|\alpha) \in F_n} f_2(t'', \alpha)$
   where f_1 and f_2 are defined as $f_1(t, \alpha) = t'$ and $f_2(t, \alpha) = \Theta$ iff $f(t, \alpha) = (t', \Theta)$.

Since the number of vectors included in A is finite and the application of f always decreases that number by one, there exists F and S for some N such that $F_N = F_{N+1} = F_{N+2} = \ldots = F$, $S_N = S_{N+1} = S_{N+2} = \ldots = S$. Then

$$\mathit{flatten}(t) \equiv (F, S)$$

Trivially t' includes no vector iff $\mathit{flatten}(t) = (t', \Theta)$. Let $\mathit{flatten}(t, \alpha) = \langle t^1|\beta^1, \ldots, t^n|\beta^n\rangle$. Then β^i's are orthogonal and t^i includes no vector.

Proposition 2 *Suppose that*

$$A = G^1 \cdot \theta^1 = G^2 \cdot \theta^2 = \cdots = G^n \cdot \theta^n$$

where θ^i's are most general unifiers and there is neither common variable between A and $G^i (i = 1, \ldots, n)$ nor between G^i and G^j $(i, j = 1, \ldots, n$ and $i \neq j)$.
It can be proved that, for $i = 1, \ldots, n$,

1. *$\theta^i = \theta^i_1 \cup \theta^i_2$, $\theta^i_1 \cap \theta^i_2 = \phi$ and*
2. *$\theta^i_1 = \{x/t \mid t$ is a non-variable term or a variable appearing in $G^i\}$ and*
3. *$\theta^i_2 = \{y/z \mid z$ is a varible appearing in $A\}$ and*
4. *for $\forall y_1/z_1, y_2/z_2$, if $z_1 = z_2$ then $y_1 = y_2$.*

θ^i_1 and θ^i_2 are called decomposition *of θ^i with respect to A.*

Definition 8 *Let U and C be a set of unifiers and a set of colors, respectively.*

$$\mathrm{compose} : S \in 2^{U \times C} \mapsto \Theta \in U$$

is defined as follows:

Let $S = \{(\theta^i, \alpha^i) \mid i = 1, \ldots, n\}$ such that $\exists A, \exists G^i$, $A = G^1 \cdot \theta^1 = G^2 \cdot \theta^2 = \cdots = G^n \cdot \theta^n$ and θ^i's are most general unifiers and there is neither common variable between A and $G^i (i = 1, \ldots, n)$ nor between G^i and G^j $(i, j = 1, \ldots, n$ and $i \neq j)$. Let θ^i_1 and θ^i_2 be decomposition of θ^i with respect to A.

$$\begin{aligned}
\Theta &= \Theta_1 \cup \Theta_2, \Theta_1 \cap \Theta_2 = \phi \\
\Theta_1 &= \theta^1_1 \cdots \theta^n_1 \\
\Theta_2 &= \{z/\langle y^{j_1}|\alpha^{j_1}, \ldots, y^{j_N}|\alpha^{j_N}\rangle \mid \forall z \in \bigcup_{i=1}^{n} \mathit{range}(\theta^i_2), \exists j_1, \ldots, j_N, y^{j_k}/z \in \theta^{j_k}_2, k = 1, \ldots, N\}
\end{aligned}$$

Θ_1 and Θ_2 are called input part and output part of Θ, respectively.

Note that if α^i's are orthgonal then vectors included in Θ are all orthogonal.

The function *compose* collects all substitutions a_k with color α_k to the same variable, X, and makes a vector substitution, $X/\langle a_1|\alpha_1, \ldots\rangle$.

4.2 Derivation

A colored world is a conjunction of colored atoms. Every colored atom $\mathcal{A}$ such that $\mathcal{A} = p(t_1, \ldots, t_n)$ and p is an or-relation is associated with a set $D(\mathcal{A})$ of clauses defining p. Let *flatten*$(\mathcal{A}) = (\langle A^1|\alpha^1, \ldots, A^n|\alpha^n\rangle, \Theta)$. Then the definition of A^i, $D(A^i)$, is defined to be equal to $D(\mathcal{A}), i = 1, \ldots, n$.

Before derivation in the vector model is described, it is worth noting the following property, which reflects that derivation is done sequentially; Every substitution made in the previous derivations has been already applied to all the constructs. Thus vectors recognized by the above *flatten* are the only vectors in the whole universe and there is no vector appearing in the future in this stage.

Definition 9 *Let $\mathcal{W}_i$ be a conjunction of colored atoms. Let $\mathcal{A}_k \in \mathcal{W}_i$ and flatten$(\mathcal{A}_k) = (\langle A_k^1|\alpha^1, \ldots, A_k^s|\alpha^s\rangle, \Theta_0)$ Assume that A_k^j is chosen for reduction by computation rule. Then derivation of $\mathcal{W}_{i+1}$ from $\mathcal{W}_i$ is defined as follows:*

Case 1 *A_k^j is unification, $X = t$:*
$\mathcal{W}_{i+1} = ((\mathcal{A}_1, \ldots, \mathcal{A}_{k-1}, \mathcal{A}_{k+1}, \ldots, \mathcal{A}_a) \cdot \Theta_0 \cdot \{X/t\}, \mathcal{A}_k - \langle A_k^j|\alpha^j\rangle)$
(Note that in $\{X/t\}$ t is not colored since it is guaranteed that X appears only in terms with the color α^j.)

Case 2 *A_k^j is an and-relation: If the following conditions are satisfied,*

1. *$\exists G_0$:- $G_1, \ldots, G_n \mid B_1, \ldots, B_m$ such that $\exists mgu\ \theta$, $A_k^j = G_0 \cdot \theta$,*
2. *$(G_1, \ldots, G_n) \cdot \theta$ has guarded refutation and generates θ',*

then $\mathcal{W}_{i+1} = ((\mathcal{A}_1, \ldots, \mathcal{A}_{k-1}, \mathcal{A}_{k+1}, \ldots, \mathcal{A}_a) \cdot \Theta_0, \langle (B_1, \ldots, B_m)|\alpha^j\rangle \cdot \theta \cdot \theta', \mathcal{A}_k - \langle A_k^j|\alpha^j\rangle)$.

Case 3 *A_k^j is an or-relation: If there exists a non-empty set $\{C_1, \ldots, C_t\} \subseteq D(A_k^j)$ such that $\exists mgu\ \theta^l$, $A_k^j = head(C_l) \cdot \theta^l, l = 1, \ldots, t$, respectively, then*

$$\mathcal{W}_{i+1} = \begin{cases} (\mathcal{A}_1, \ldots, \mathcal{A}_{k-1}, \mathcal{A}_{k+1}, \ldots, \mathcal{A}_a) \cdot \Theta_0 \cdot \Theta_2, & \textit{if } D_s = \phi \\ (\langle body(C_1)|\alpha^{j1}, \ldots, body(C_t)|\alpha^{jt}\rangle) \cdot \Theta_1, & \\ \mathcal{A}_k - \langle A_k^j|\alpha^j\rangle) & \\ & \\ (\mathcal{A}_1, \ldots, \mathcal{A}_{k-1}, \mathcal{A}_{k+1}, \ldots, \mathcal{A}_a) \cdot \Theta_0 \cdot \Theta_2, & \textit{otherwise} \\ (\langle body(C_1)|\alpha^{j1}, \ldots, body(C_t)|\alpha^{jt}\rangle) \cdot \Theta_1, & \\ \langle A_k^1|\alpha^1, \ldots, A_k^{j-1}|\alpha^{j-1}, (\overline{A}_k^j|_{D_s})|\alpha^j, A_k^{j+1}|\alpha^{j+1}, \ldots, A_k^s|\alpha^s\rangle & \end{cases}$$

where

- *$\overline{A}_k^j$ is a variant of A_k^j.*
- *$\overline{A}_k^j|_{D_s}$ and $A_k^j|_{D_s}$ are atoms with a definition D_s where $D_s = D(A_k^j) - D_f - \{C_1, \ldots, C_t\}$ and $D_f \subset D(A_k^j)$ is a set of clauses which are not unifiable with A_k^j.*
- *Θ_1 and Θ_2 are input and output parts of Θ, respectively,*
$\Theta = compose(\{(\theta^j, \alpha^{jl}) \mid l = 1, \ldots, t'\})$ where α^{jl}, $l = 1, \ldots, t'$, are new colors such

that $\alpha^j \subset \alpha^{jl}, \alpha^{jl_1} \perp \alpha^{jl_2}$ *if* $l_1 \neq l_2$.
$t' = t$ *if* $D_s = \phi$, *otherwise* $t' = t+1$.
θ^{t+1} *is mgu such that* $A_k^j = \overline{A_k^j} \cdot \theta^{t+1}$.

Example 1 *Suppose that* $\mathcal{W}_i = \mathcal{A}_j, q(Y)$ *where* $\mathcal{A}_j = p(\langle 1|\alpha^1, 2|\alpha^2\rangle, Y)$. *Assume that p is an and-relation and its definition is*

$$p(1, X)\ :\text{-}\ true \mid r_1(X)$$

$$p(2, X)\ :\text{-}\ true \mid r_2(X)$$

Then

$$\Theta \equiv \{Y/\langle X_1|\alpha^1, X_2|\alpha^2\rangle\}$$
$$\mathcal{W}_{i+1} \equiv r_1(X_1), r_2(X_2), q(\langle X_1|\alpha^1, X_2|\alpha^2\rangle)$$

Example 2 *Suppose that* $\mathcal{W}_0 = n\text{-}merge([a,b], V, V)$, *where n-merge is an or-relation with the following definition.*

$$\begin{aligned}
C_1 &: n\text{-}merge([A^1|X^1], Y^1, Z^1) : -Z^1 = [A^1|ZZ^1], n\text{-}merge(X^1, Y^1, ZZ^1)\\
C_2 &: n\text{-}merge(X^2, [A^2|Y^2], Z^2) : -Z^2 = [A^2|ZZ^2], n\text{-}merge(X^2, Y^2, ZZ^2)\\
C_3 &: n\text{-}merge([], Y^3, Z^3) : -Z^3 = Y^3\\
C_4 &: n\text{-}merge(X^4, [], Z^4) : -Z^4 = X^4
\end{aligned}$$

Only C_1 *can have successful head unification.*

$$\begin{aligned}
D_f &= \{C_3\}, D_s = \{C_2, C_4\}\\
\theta &= \{A_1^1/a, X_1^1/[b], Y_1^1/Z_1^1, Z_1^1/V\}\\
\Theta_1 &= \{A_1^1/a, X_1^1/[b], Y_1^1/Z_1^1\}\\
\Theta_2 &= \{V/\langle Z_1^1|\alpha^1, V'|\alpha^2\rangle\}
\end{aligned}$$

Then $\mathcal{W}_1$ *is:*

$$\mathcal{W}_1 = \langle (Z_1^1 = [a|ZZ_1^1], n\text{-}merge([b], Z_1^1, ZZ_1^1))|\alpha^1\rangle, n\text{-}merge([a,b], V', V')|_{D_s}$$

If a unification atom is reduced, then $\mathcal{W}_2$ *is:*

$$\mathcal{W}_2 = \langle n\text{-}merge([b], [a|ZZ_1^1], ZZ_1^1)|\alpha^1\rangle, n\text{-}merge([a,b], V', V')|_{D_s}$$

Suppose that the first atom is chosen for reduction. C_1 *and* C_2 *have successful head unification.*

$$\begin{aligned}
D_f &= \{C_3, C_4\}, D_s = \phi\\
\theta^1 &= \{A_2^1/a, X_2^1/[], Y_2^1/[a|ZZ_1^1], Z_2^1/ZZ_1^1\}\\
\theta^2 &= \{A_2^2/a, X_2^2/[b], Y_2^2/[ZZ_1^1], Z_2^2/ZZ_1^1\}\\
\Theta_1 &= \{A_2^1/b, X_2^1/[], Y_2^1/[a|ZZ_1^1], A_2^2/a, X_2^2/[b], Y_2^2/[ZZ_1^1]\}\\
\Theta_2 &= \{ZZ_1^1/\langle Z_2^1|\beta^1, Z_2^2|\beta^2\rangle\}
\end{aligned}$$

$\mathcal{W}_3$ is:

$$\begin{aligned}\mathcal{W}_3 \;=\;& n\text{-}merge([a,b],V',V')|_{D},\\ & \langle (Z_2^1=[b|ZZ_2^1], n\text{-}merge([],[a|ZZ_1^1],ZZ_2^1))|\beta^1 (Z_2^2=[a|ZZ_2^2], n\text{-}merge([b],[ZZ_1^1],ZZ_2^2))|\beta^2\rangle.\end{aligned}$$

After further reduction of unification atoms, V becomes the following vector:

$$V = \langle [a,b|ZZ_2^1]|\beta^1 \cup \alpha^1, [a,a|ZZ_2^2]|\beta^2 \cup \alpha^1, V'|\alpha^2\rangle$$

5 Comparison

Branching world model is eager copying scheme. When an or-relation is reduced, copies of the whole world are created at once. The overhead of handling or-relations is in making copies of a world.

On the contrary, vector model is lazy copying scheme. When an or-relation is reduced using, say n clauses, only variables are copied into a vector of size n, which are propagated to other atoms sharing the same variables. Such a vector will invoke further copying by *flatten* when an atom including the vector is selected for reduction. In this way, copies of a world are created part by part incrementally. In this model, the overhead is in *flatten*, that is, partial copying.

Another important difference is that the number of reductions in the vector model is less than that in the branching model. It is explained using an example. Suppose that there are three atoms,

$$p(X,Y), q(X), r(Y)$$

where p is an and-relation which suspends until its two arguments will be instantiated, and q and r are or-relations. Suppose also that q and r have the following definitions.

$$\begin{aligned} q(X) &\;:- X = a_1.\\ &\;\;\vdots\\ q(X) &\;:- X = a_n.\\[1em] r(Y) &\;:- Y = b_1.\\ &\;\;\vdots\\ r(Y) &\;:- Y = b_m.\end{aligned}$$

In the branching world model, if q is executed first, r is executed n times. If r is executed first, q is executed m times. However, in the vector model, q and r are executed once. Suppose that q is executed first. q generates colored unifier $\{X/\langle a_1|\alpha^1,\ldots,a_n|\alpha^n\rangle\}$ and terminates. Thus the goals become:

$$p(\langle a_1|\alpha^1,\ldots,a_n|\alpha^n\rangle, Y), r(Y)$$

Since p waits for values at the second argument, r is the only atom that can be reduced next. Making colored unifier $\{Y/\langle b_1|\beta^1,\ldots,b_m|\beta^m\rangle\}$, the execution of r terminates. The goals becomes:

$$p(\langle a_1|\alpha^1,\ldots,a_n|\alpha^n\rangle, \langle b_1|\beta^1,\ldots,b_m|\beta^m\rangle)$$

The whole computation terminates after executing p $n \times m$ times. Clearly q and r are executed only once. This is because q and r invoke copying of only relevant (sharing variables) goals. The difference is apparent when goals are $q(X), r(Y)$. In the vector model, both goals are executed only once, while in the branching world model q is executed m times or r is executed n times.

The advantage of the vector model with respect to the number of reductions comes from the nature of the lazy copying. In the lazy copying, copying is performed part by part incrementally. Therefore result of computation performed in a position which is not yet copied will be shared later by its copies. The advantage of the vector model is amplified when number of goals are large and there are many independent sources of or-parallel branching.

6 Optimization/Parallalizing the vector model

In spite of the advantages of the vector model, its complexity of handling vectors can not be ignorable. Two operations can be considered as sources of its complexity. One is *flatten*, and the other is *compose*. Let us consider them in detail.

Flattening of a colored atom is needed in order to avoid unification between a vector and an ordinary term during head unification. *However flattening of a vector deeply embedded in a term is not necessary if it is not unified with an ordinary term.* In fact a deeply embedded vector does not affect result of head unification. Even if it is flattened and n atoms are generated, n head unification do the same thing. Furthermore, a deeply embedded vector will be eventually flattened when needed. Thus it is advantageous to avoid flattening and to share result of computation. This is equivalent to that worlds embodied in a vector share the same result of computation without iterating the same computation. This leads to an idea of *bounded depth flattening*.

Another point of *flatten* is construction of Θ_0. When a colored atom is flattened with respect to a vector, variables outside the vector have to be collected together to make Θ_0. The complexity of this is proposional to the size of an atom. The bounded depth flattening never helps its reduction. *However if a variable outside the vector is known not to be instantiated during the computation of the atom*[2], *it is not necessary to take the variable into account, since the purpose of Θ_0 is to prepare for multiple assignment to a variable in a goal atom.* Variables taken into account are those which are not known to be reference-only. This optimization becomes more feasible by the introduction of *mode declaration.* For each and- and or-relation, one mode is defined. A mode specifies, for each argument, that it is *referece-only (read)* or *not (write).* Note that an argument with "write" mode need not be instantiated to a non-variable term. In order to enjoy full advantage of mode declaration, *single producer constraint* must be guranteed. The single producer constraint means that at most one occurence of a variable can appear at argument position with "write" mode. If this constraint is guaranteed, it is guaranteed that a term appearing at an argument position with "write" mode is always an unbound variable or a vector of unbound variables. Owing to a mode declaration and the single producer constraint, construction of Θ_0 becomes statically predictable.

compose is performed only in the reduction of an or-relation. Complexity of *compose* is in construction of Θ_2. The purpose of Θ_2 is the preparation of multiple instantiation of a variable

[2]By the term "computation of an atom", we mean the whole tree of computation initiated by that atom.

in a goal atom by multiple clauses as in the case of Θ_0. In general, all the variables in a goal atom must be collected to construct Θ_2. The introduction of mode declaration and the single producer constraint reduces the complexity also in this case. Under these restrictions, Θ_2 can be constructed from terms appearing in argument positions with "write" mode.

And-parallel computation and or-parallel computation of logic programs have been investigated indenpendently for many years. Experience for realizations of each parallelism has been accumulated by many researchers. Outcomes of these researches are Or-parallel Prologs and committed choice languages. Or-parallel Prologs exploit or-parallelism with and-sequential computation, while committed choice languages exploit and-parallelism with or-sequential or shallow or-parallel computation.

Recently mixing both parallelism takes much attention and is being investigated intensively [Yang], [Haridi], [Naish]. When combining them, the most important design issues are 1) to exploit both parallelism without restricting either, and 2) to make the size of atomic operations needed for combination as small as possible. The vector model, takes an approach characterized as "eager branching and lazy copying". Eager branching implies that "branch whenever you can", thus it exploits full or-parallelism. Lazy copying contributes to reduce the size of such an atomic operation as copying of a world. In lazy copying scheme, computation of an atom can proceed without waiting for copying of the whole world being completed, while in eager copying scheme it must wait. In Section 4 the vector model was described assuming sequential derivation. In other words, the entire derivation was assumed to be an atomic operation. The reason why sequentiality is needed is to guarantee that once a colored atom is flattened no vector appears until next derivation, thus to avoid collision of a vector and an ordinary term during head unification. However, as discussed above, head unification only needs information within bounded depth, therefore it is enough to flatten a colored atom to that depth. Owing to that, an atomic operation is derived from the entire derivation into smaller operations such as bounded depth *flatten* and *compose*. Again, as we discussed previously, atomicities of operations can be reduced by two language restrictions, mode declaration and the single producer constraint. Under these restrictions, the sizes of atomicity required by these operations becomes those required by a unification operation.

7 CCL implementation

One of the main targets of ANDOR-II project is to find a class of and- or-parallel languages that can be efficiently implemented in CCL. The language and its computation model explained so far are almost equivalent to actual ANDOR-II and its model. In this section, we briefly explain how it is implemented in GHC, one of CCL. For more detail, see other paper[Takahashi].

An ANDOR-II program is translated into a GHC program, which includes all codes realizing such operations as *flatten* and *compose*. Current translation strategy is characterized by the followings.

- Vector/scalar analysis of arguments: For each clause, vector/scalar analysis determines, for each argument of an atom in body part, that it is of a vector type or of a scalar type, which means an ordinary term. The analysis is guided by modes of atoms and utilizing the facts that an output of an or-relation is always of a vector type, and that an output type of an and-relation is of a scalar type if all input types are of scalar, otherwise it is a vector.

- Lifting a vector to an outermost level: A unification atom is the only operation that has the possibility to instantiate a variable to a term with a vector in it. In ANDOR-II/CCL implementation, such unifications are detected statically and replaced by unifications that lifts a vector to an outermost level as follows. First for each variable appearing in it, mode and a type (vector/scalar) are determined. When an unbound variable is to be instantiated to a term which includes a variable with mode reference-only and a type vector, the term is flattened first and the resultant vector of flattened terms is assigned to the variable. Note that it implies that a term is never shipped out until the vector arrives. This is not the case of a term with a variable which is never instantiated, that is, has no producer.
- Special treatment of lists: A unification, $X = [A|Y]$, is treated exceptionally if X's mode is "write", A's mode is "reference-only", and mode and a type of Y is "reference-only" and a vector, respectively. In this case, as an exception to the above, a cons cell is shipped out without flattening with respect to Y while A must be flattened if it is a vector. Such an early ship out is needed to enable stream programming.

These three are coupled with each other. The followings are worth noting:

1. Since a vector is lifted to an outermost level, flattening is enough to be done only in the first level.
2. Since argument types are determined statically a code for flattening can be placed at an appropriate position in a translated program.

In this strategy, vector/scalar analysis plays a central role. However, special treatment of lists makes it troublesome. We are now trying to clarify it. Also we are now considering a new translation strategy which does not depend on vector/scalar analysis.

References

[Haridi] S. Haridi, P. Brand, *ANDORRA Prolog -- An Integration of Prolog and Committed Choice Languages*, Proc. of FGCS'88, 1988, pp.745-754.

[Naish] L. Naish, *Parallelizing NU-Prolog*, Proc. of Int. Conf. Logic Programming, 1988, pp.1546-1564.

[Takahashi] K. Takahashi, A. Takeuchi, T. Yasui, *A Parallel Problem Solving Language ANDOR-II and its Implementation*, to appear ICOT TR.

[Takeuchi] A. Takeuchi, K. Takahashi, H. Shimizu, *A Parallel Problem Solving Language for Concurrent Systems*, ICOT TR-418, 1988. Also to appear in *Concepts and Characteristics of Knowledge-based Systems*, M. Tokoro, Y. Anzai, A. Yonozawa (eds.), North-Holland.

[Ueda] K. Ueda, *Guarded Horn Clauses: A Parallel Logic Programming Language with the Concept of a Guard*, ICOT TR-208, 1986.

[Yang] R. Yang, H. Aiso, *P-Prolog: A Parallel Logic Language based on Exclusive Relation*, Proc. of 3rd Int. Conf. Logic Programming, 1986, pp.255-269.