Fundamenta Informaticae XIII (1990) 67-83
IOS press

ON DOWNWARD CLOSURE ORDINALS OF LOGIC PROGRAMS

Rajiv BAGAI,* Marc BEZEM,** and M.H. van EMDEN*

* *Dept. of Computer Science, Univeristy of Victoria, Victoria, BC V8W 2Y2, Canada*
***CWI, Kruislaan 413, 1098 SJ Amsterdam, Netherlands*

Abstract

Blair has shown that for every ordinal up to and including the least non-recursive ordinal there exists a logic program having that ordinal as downward closure ordinal. However, given such an ordinal and Blair's proof, it is not straightforward to find a corresponding logic program. In fact, in the literature only a few isolated, *ad hoc*, examples of logic programs with downward closure ordinal greater than ω can be found. We contribute to bridging the gap between what is known abstractly and what is known concretely by showing the connection between some of the existing examples and the well-known concept of the order of a vertex in a graph. Using this connection as a basis, we construct a family $\{P_\alpha\}_{\alpha<\epsilon_0}$ of logic programs where any member P_α has downward closure ordinal $\omega + \alpha$.

1 Introduction

The functions or relations computed by programs are usually characterized mathematically by associating a certain mapping with each program. What is computed can then be regarded as a fixpoint of that mapping. In logic programming such mappings and their fixpoints play a predominant role in the semantical discussions (see for example [2] and [6]). The fixpoints are subject to an order, so that we can distinguish the greatest and the least fixpoints as being of particular interest. The least fixpoint characterizes the positive information contained in a logic program; the greatest fixpoint bears some relationship to negative information contained in the program. It is beyond the scope of this paper to explain this latter relationship in technical terms. The interested reader is referred to the literature mentioned above.

The least fixpoint is a limit of all finite powers of the mapping. This is not the case for the greatest fixpoint. However, when we generalize the notion of power to include transfinite powers, we find that the greatest fixpoint can also be characterized as a limit of powers. In this more general setting, we call the least power for which the least (greatest) fixpoint is reached the upward (downward) closure ordinal. The lack of symmetry between the fixpoints that we just referred to can then be expressed by saying that the upward closure ordinal is at most ω, and that the downward closure ordinal can be greater than ω for certain logic programs. In fact, Blair has shown in [6] that for every ordinal up to and including the least non-recursive ordinal (ω_1^{ck}), there is a logic program having that ordinal as downward closure ordinal. However, given such an ordinal and Blair's proof, it is not

*Dept. of Computer Science, University of Victoria, Victoria B.C., Canada V8W 2Y2
†C.W.I., Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

straightforward to find a logic program having this ordinal as downward closure ordinal. There is a painful contrast between the richness, *in abstracto*, assured by Blair's theorem and the meagreness of what is known concretely: the literature presents only a few isolated examples of logic programs with downward closure ordinal greater than ω, presented in an *ad hoc* manner. It is the purpose of our paper to soften this contrast.

The basis of our approach is provided by the well-known notion of ordering vertices of a directed graph by assigning an ordinal number, which may or may not be finite, to each vertex from which no infinite path starts. The connection between graphs and logic programs is provided by a variant of Kowalski's reachability representation of graphs. These fundamentals allow us to "explain" some of the published examples of logic programs having downward closure ordinal exceeding ω. More importantly, they suggest a family of logic programs having as downward closure ordinals all those ordinals for which a certain convenient notation system applies.

In Section 2 we review some of the theory on fixpoints and closure ordinals. In the next section we explain some of the examples by relating them to known concepts in graph theory. In Section 4 we prepare for a more general treatment by exploiting generally applicable representations of graphs by means of logic programs. Section 5 is devoted to the construction of a family of logic programs indexed by ordinals up to ϵ_0, the least fixpoint of the function $\lambda\alpha[\omega^\alpha]$. Each member P_α of this family represents a graph, and has $\omega + \alpha$ as downward closure ordinal.

2 Fixpoints and closure ordinals

Throughout this paper we shall assume knowledge of Lloyd [8], but for convenience of the reader we recall some definitions and results which are frequently used in the sequel.

Let B_P be the Herbrand Base (the set of all variable-free atoms) corresponding to a logic program P. The *immediate-consequence function* $T_P : \mathcal{P}(B_P) \to \mathcal{P}(B_P)$ associated with P, mapping Herbrand interpretations I to Herbrand interpretations $T_P(I)$, is defined by: $A \in T_P(I)$ iff there exists a variable-free instance

$$A \leftarrow B_1 \& \ldots \& B_n \quad (n \geq 0)$$

of a clause in P such that $\{B_1, \ldots, B_n\} \subseteq I$.

As shown by van Emden and Kowalski in [11] and by Apt and van Emden in [2], the immediate-consequence function, being monotonic and continuous in the complete lattice $(\mathcal{P}(B_P), \subseteq)$, has a unique least fixpoint (denoted $\mathrm{lfp}(T_P)$) and a unique greatest fixpoint (denoted $\mathrm{gfp}(T_P)$). Approximations to one of the fixpoints are obtained by means of a, possibly transfinite, sequence of sets having the fixpoint as limit.

Since T_P is monotonic with respect to set inclusion, the sequence $T_P^n(\emptyset)$, where n runs through the natural numbers (i.e. the finite ordinals) is nondecreasing. The limit of this sequence is its union, which is the least fixpoint of T_P.

Dually, the sequence $T_P^n(B_P)$ is nonincreasing and the limit of this sequence is its intersection, which is *not* necessarily a fixpoint of T_P, as the following example from [8] shows.

Example 1 Let P be the program

```
p(s(X)) <- p(X);
q(0) <- p(X);
```

follow Prolog's convention of identifiers starting with upper-case letters for variables. an be seen that for $n > 0$,

$$T_P^n(B_P) = \{\mathtt{q(0)}\} \cup \{\mathtt{p(s}^m\mathtt{(0))} | m \geq n\}.$$

e intersection of $T_P^n(B_P)$ for all finite n is $\{\mathtt{q(0)}\}$. This is not a fixpoint, since $T_P(\{\mathtt{q(0)}\})$).

his apparent breakdown of duality is puzzling. To better understand what is going on, sequences $T_P^n(\emptyset)$ and $T_P^n(B_P)$ are generalized according to the following definition of *ordinal powers* of T_P:

$$\begin{aligned} T_P \uparrow 0 &= \emptyset, \\ T_P \uparrow \alpha &= T_P(T_P \uparrow (\alpha - 1)), \quad \text{if } \alpha \text{ is a successor ordinal;} \\ &= \bigcup\{T_P \uparrow \beta | \beta < \alpha\}, \quad \text{if } \alpha \text{ is a limit ordinal.} \end{aligned}$$

ally,

$$\begin{aligned} T_P \downarrow 0 &= B_P, \\ T_P \downarrow \alpha &= T_P(T_P \downarrow (\alpha - 1)), \quad \text{if } \alpha \text{ is a successor ordinal;} \\ &= \bigcap\{T_P \downarrow \beta | \beta < \alpha\}, \quad \text{if } \alpha \text{ is a limit ordinal.} \end{aligned}$$

ote that for finite ordinals n, $T_P \uparrow n = T_P^n(\emptyset)$ and $T_P \downarrow n = T_P^n(B_P)$.

efinition The *upward closure ordinal* of T_P is the least ordinal α such that $T_P \uparrow \alpha = (T_P)$; the *downward closure ordinal* of T_P, denoted $\mathrm{dco}(T_P)$, is the least ordinal α such at $T_P \downarrow \alpha = \mathrm{gfp}(T_P)$.

As shown in [2], both the upward and downward closure ordinals of T_P exist for any logic ogram P. Moreover, the upward closure ordinal is at most ω (which follows essentially om the fact that the right hand sides of clauses in a logic program are finite). The st published example of a logic program whose downward closure ordinal exceeds ω reproduced as Example 3 below. It was published in [2] and has been attributed in part K.L. Clark and in part to H. Andreka and I. Nemeti. For the program of Example 1, e downward closure ordinal is $\omega + 1$. On the other hand, certain classes of programs ve downward closure ordinals of at most ω; for example, if in every clause the conclusion ntains all variables that occur in the clause.

For any program P, it is the case that $T_P \uparrow \alpha \subseteq T_P \downarrow \beta$, for all α and β. An important ss is that of determinate programs as defined in Blair [6]:

efinition A program P is *determinate* if $T_P \uparrow \omega = T_P \downarrow \omega$.

roposition 1 *If P is determinate, then $dco(T_P) \leq \omega$.*

roof For any P, we have that $T_P \uparrow \omega = \mathrm{lfp}(T_P) \subseteq \mathrm{gfp}(T_P) \subseteq T_P \downarrow \omega$. Thus determinacy plies that $T_P \downarrow \omega$ is a (in fact, the only) fixpoint. The proposition then follows from the efinition of dco. □

Determinacy of a program is a fixpoint-theoretic property. Using the notion of SLD-erivation we define a stronger, proof-theoretic, property of programs, which serves as an termediate step towards a convenient sufficient condition for determinacy.

Definition A program is *terminating* if no infinite SLD-derivation starts from a variable-free goal.

Proposition 2 *Every terminating program is determinate.*

Proof Let P be a terminating program and $A \in B_P$, i.e. A is any variable-free atom. Since any SLD-derivation starting from the goal $\leftarrow A$ is finite, any SLD-tree with $\leftarrow A$ as the root is either finitely failed or contains a successful derivation. That is, A is either in the finite-failure set (equal to $B_P \setminus T_P \downarrow \omega$, see [2]) or in the success set (equal to $T_P \uparrow \omega$, see [2]). Therefore, $T_P \uparrow \omega = T_P \downarrow \omega$. □

As it is possible for a variable-free negative clause $\leftarrow A$ to begin a successful as well as an infinite SLD-derivation, the converse of Proposition 2 does not hold in general. For example, consider the program P:

```
q;
q <- q;
```

This program is determinate because

$$T_P \uparrow \omega = \{\mathtt{q}\} = T_P \downarrow \omega.$$

However, P is not terminating since the variable-free clause `<- q` heads an infinite SLD-derivation, namely

$$\mathtt{<-\ q}, \mathtt{<-\ q}, \mathtt{<-\ q}, \ldots.$$

The following result, combined with Proposition 2, yields the desired sufficient condition for determinacy. This result may be seen as a precursor of Theorem 2.8 from Bezem [4].

Proposition 3 *Let P be a logic program such that all its predicate symbols have non-zero arity and every term occurring in the body of any of its clauses is a proper subterm of a term occurring in the head of the same clause. Then P is terminating.*

Proof Straightforward by structural induction on the terms occurring in any variable-free goal. □

The relation between a program's structure and its downward closure ordinal is not well understood. In the literature, a few isolated examples are exhibited as curiosities to show that the value of this function can exceed ω. Let us discuss some other examples found in Lloyd [8].

Example 2 Let P be the program

```
p(f(X)) <- p(X);
q(a) <- p(X);
q(f(X)) <- q(X);
r(a) <- q(X);
r(f(X)) <- r(X);
s(a) <- r(X);
s(f(X)) <- s(X);
t(a) <- s(X);
t(f(X)) <- t(X);
```

Then we have

$$\begin{aligned} T_P \downarrow 0 &= B_P, \\ T_P \downarrow \omega &= T_P \downarrow 0 \setminus \{\mathtt{p}(\mathtt{f}^k(\mathtt{a}))|k < \omega\}, \\ T_P \downarrow \omega 2 &= T_P \downarrow \omega \setminus \{\mathtt{q}(\mathtt{f}^k(\mathtt{a}))|k < \omega\}, \\ T_P \downarrow \omega 3 &= T_P \downarrow \omega 2 \setminus \{\mathtt{r}(\mathtt{f}^k(\mathtt{a}))|k < \omega\}, \\ T_P \downarrow \omega 4 &= T_P \downarrow \omega 3 \setminus \{\mathtt{s}(\mathtt{f}^k(\mathtt{a}))|k < \omega\}, \\ T_P \downarrow \omega 5 &= T_P \downarrow \omega 4 \setminus \{\mathtt{t}(\mathtt{f}^k(\mathtt{a}))|k < \omega\}, \\ &= \emptyset, \\ &= \mathrm{gfp}(T_P). \end{aligned}$$

Thus $\mathrm{dco}(T_P)$ is $\omega 5$, since that is the least ordinal α such that $T_P \downarrow \alpha = \mathrm{gfp}(T_P)$.

Example 3 Let P be the program

```
p(a) <- p(X) & q(X);
p(f(X)) <- p(X);
q(b);
q(f(X)) <- q(X);
```

Then we have

$$\begin{aligned} T_P \downarrow n &= \{\mathtt{p}(\mathtt{f}^k(\mathtt{a}))|k < \omega\} \cup \{\mathtt{p}(\mathtt{f}^k(\mathtt{b}))|n \leq k < \omega\} \cup \\ &\quad \{\mathtt{q}(\mathtt{f}^k(\mathtt{a}))|n \leq k < \omega\} \cup \{\mathtt{q}(\mathtt{f}^k(\mathtt{b}))|k < \omega\}, \quad \text{for } n < \omega, \\ T_P \downarrow \omega &= \{\mathtt{p}(\mathtt{f}^k(\mathtt{a}))|k < \omega\} \cup \{\mathtt{q}(\mathtt{f}^k(\mathtt{b}))|k < \omega\}, \\ T_P \downarrow (\omega + n) &= \{\mathtt{p}(\mathtt{f}^k(\mathtt{a}))|n \leq k < \omega\} \cup \{\mathtt{q}(\mathtt{f}^k(\mathtt{b}))|k < \omega\}, \quad \text{for } n < \omega, \\ T_P \downarrow \omega 2 &= \{\mathtt{q}(\mathtt{f}^k(\mathtt{b}))|k < \omega\}, \\ &= \mathrm{gfp}(T_P). \end{aligned}$$

The above shows that $\mathrm{dco}(T_P)$ is $\omega 2$.

Example 4 Let P be the program

```
p(a) <- p(X);
p(f(X)) <- p(X);
q(b);
q(f(X)) <- q(X);
r(c) <- r(X) & q(X);
r(f(X)) <- r(X);
```

Then $\mathrm{dco}(T_P)$ is easily shown to be $\omega 2$. Lloyd [8] is really scraping the bottom of the barrel here: if we remove the clauses for the predicate symbol p, which do not affect $\mathrm{dco}(T_P)$, then we obtain the program of Example 3, up to renaming of symbols.

3 Graphs associated with uniconditional logic programs

We have seen some examples of programs with downward closure ordinal greater than ω. With these in mind one can, with a bit of tinkering, produce more. But that exercise may

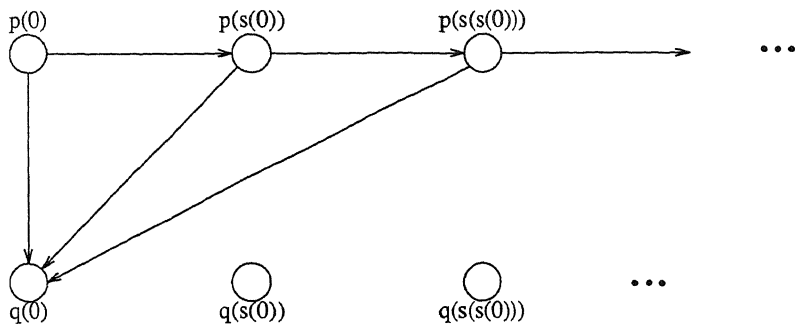

Figure 1: Graph of Example 1

not make clear what the *mechanism* is; *why* the examples work. In this section we show that uniconditional logic programs, that is, those where every clause has one condition, can be associated with directed graphs in a natural way. From graph theory we know how to assign ordinals to vertices of a directed graph. A direct relationship will be established between the downward closure ordinal of the logic program and (the dual of) this ordinal assignment. As it is easier to construct graphs in such a way that all successive ordinals up to a certain transfinite bound are assigned to some vertex, this result suggests a way to construct *ad libitum* examples of logic programs with downward closure ordinal beyond ω.

Of course, we do not suggest that this be actually done. We present this result because it substantiates our claim that we now *understand* some of the published examples. Better still is to have a parameterized family of logic programs with a downward closure ordinal closely related to the parameter. This parameter is given as a term encoding ordinal numbers up to a certain bound. That will be the topic of the next two sections.

The graph *associated with* a uniconditional logic program has as vertices elements of the Herbrand base, that is, ground atomic formulas constructed with symbols occurring in the program. There is an arc from `A` to `B` iff `B <- A` is a variable-free instance of a clause in the program. In Figure 1 we show the graph associated with Example 1 discussed in the previous section. We will now review known concepts in graph theory that relate directly to the downward closure ordinal of a uniconditional program.

Let $G = \langle V, E \rangle$ be a directed graph, where V is a (possibly infinite) set of vertices and $E \subseteq V \times V$ is a set of edges. The inverse graph of G, denoted G^{-1} is the graph G with all edges reversed, i.e. $G^{-1} = \langle V, E^{-1} \rangle$.

Let $R_G : \mathcal{P}(V) \rightarrow \mathcal{P}(V)$ be defined as

$$R_G(X) = \{v | \exists u \in X : \langle u, v \rangle \in E\}.$$

We call R_G the *reachability function* of G, because $R_G(X)$ is the set of vertices reachable in one step from a vertex in X. Clearly, R_G is monotonic and we have the following proposition.

Proposition 4 *If G is the graph associated with a uniconditional logic program P, then R_G is the immediate-consequence function T_P.*

Definition The *upward ordinal function* X_G for the graph G maps ordinals to sets of vertices of G as follows:

$$\begin{aligned} X_G(0) &= \emptyset, \\ X_G(\alpha) &= \{x | R_G(\{x\}) \subseteq X_G(\alpha - 1)\}, \quad \text{if } \alpha \text{ is a successor ordinal;} \\ &= \bigcup\{X_G(\beta) | \beta < \alpha\}, \quad \text{if } \alpha \text{ is a limit ordinal.} \end{aligned}$$

A similar function is defined in Berge [3], where it is called "ordinal function". Recall that a well-founded partial ordering is an irreflexive, transitive relation allowing no infinite descending sequences. Informally, X_G gives a transfinite construction for the largest well-founded partial ordering which can be embedded in the transitive closure of G.

We find it useful to define the dual of Berge's upward ordinal function:

Definition The *downward ordinal function* Y_G for the graph $G = \langle V, E \rangle$ is defined as follows:

$$\begin{aligned} Y_G(0) &= V, \\ Y_G(\alpha) &= R_G(Y_G(\alpha - 1)), \quad \text{if } \alpha \text{ is a successor ordinal;} \\ &= \bigcap\{Y_G(\beta) | \beta < \alpha\}, \quad \text{if } \alpha \text{ is a limit ordinal.} \end{aligned}$$

The following two propositions are straightforward.

Proposition 5 *For all α, $Y_G(\alpha) = V \setminus X_{G^{-1}}(\alpha)$.*

Proposition 6 *Y_G is non-increasing, i.e. if $\alpha \leq \beta$, then $Y_G(\beta) \subseteq Y_G(\alpha)$.*

Proposition 7 *Let $S \subseteq V$ be such that $S \subseteq R_G(S)$. Then $S \subseteq Y_G(\alpha)$, for all α.*

Proof Clearly $S \subseteq Y_G(0)$. Now suppose $S \subseteq Y_G(\beta)$, for all $\beta < \alpha$. If α is a successor ordinal then by the assumption on S and monotonicity of R_G we have $S \subseteq R_G(S) \subseteq R_G(Y_G(\alpha - 1)) = Y_G(\alpha)$. If α is a limit ordinal, the result follows from the definition of $Y_G(\alpha)$. □

Corollary If a vertex x occurs in a cycle, then $x \in Y_G(\alpha)$, for all α.

Proof For any cycle S in V we have $S \subseteq R_G(S)$. □

Definition The *upward order* of a vertex x is defined in Berge [3] as the smallest ordinal α such that $x \in X_G(\alpha)$, provided that there exists such α.

The above definition assigns an order to vertices from which no infinite path starts (cf. [3, p. 21]). Note that upward orders can only be successor ordinals.

Before we can develop a notion dual to the upward order, we need the following result:

Proposition 8 *For any $x \in V$, if there is an ordinal α such that $x \notin Y_G(\alpha)$, then there is a greatest ordinal β such that $x \in Y_G(\beta)$.*

Proof If $x \notin Y_G(\alpha)$ then by the well-ordering property of ordinals there exists a least ordinal α' such that $x \notin Y_G(\alpha')$. We know that $\alpha' \neq 0$ since $Y_G(0) = V$; so α' must be either a successor or a limit ordinal. The latter case can be excluded since then by the definition $Y_G(\alpha') = \bigcap\{Y_G(\beta) | \beta < \alpha'\}$, we have that $x \notin Y_G(\alpha')$ implies $x \notin Y_G(\beta)$ for some $\beta < \alpha'$, contradicting the minimality of α'. Thus the only possible case turns out to be $\alpha' = \beta + 1$ for some β, which is the desired maximal solution of $x \in Y_G(\beta)$. □

Definition The *downward order* of a vertex x is the greatest ordinal β such that $x \in Y_G(\beta)$, provided that there is an α such that $x \notin Y_G(\alpha)$.

Proposition 9 *Let $D(x)$ be the downward order of x in G and $U(x)$ be the upward order of x in G^{-1}. Then $U(x) = D(x) + 1$ for every $x \in V$ such that $x \notin Y_G(\alpha)$ for some α.*

Proof Follows from Proposition 5. □

In the remaining part of the paper, by *order* we mean the downward order of a vertex. We also denote this value by $Order(x)$, for any vertex x having a downward order.

For certain graphs, *Order* fails to be a total function; for instance it is not defined for vertices in a cycle since such vertices occur in $Y_G(\alpha)$, for all ordinals α. Moreover, the range of *Order* is an initial segment of the ordinals, i.e. for any ordinal α if there is a vertex $u \in V$ such that $Order(u) = \alpha$, then for every ordinal $\beta < \alpha$ there is a vertex $v \in V$ such that $Order(v) = \beta$.

Definition For any vertex x of G, let $x^{\leftarrow}$ be the set of all vertices from which there is an edge to x.

The following proposition is a simple consequence of the definitions.

Proposition 10 *If all vertices in $x^{\leftarrow}$ have an order, then the order of x is the least ordinal greater than all orders of vertices in $x^{\leftarrow}$.*

Definition A *finitary* path in a graph is a path on which every vertex occurs at most a finite number of times. A graph G is *finitarily anti-founded* if G^{-1} does not contain any infinite finitary paths.

Proposition 11 *For any vertex x of a finitarily anti-founded graph, $Order(x)$ is defined iff there is no vertex y such that y occurs in a cycle and there is a path from y to x.*

Proof ($\Rightarrow$) Suppose such a vertex y exists. Then there is a cycle C containing y and a path P from y to x. Clearly, $x \in C \cup P \subseteq R_G(C \cup P)$. By Proposition 7, $x \in Y_G(\alpha)$, for all α. Thus $Order(x)$ is not defined.

($\Leftarrow$) Suppose $Order(x)$ is not defined. We have to show that there exists a vertex y such that y occurs in a cycle and there is a path from y to x. Let y_0 be x. By Proposition 10, there is a vertex $y_1 \in y_0^{\leftarrow}$ such that $Order(y_1)$ is not defined. By iterating the same argument we get an infinite path $\langle y_0, y_1, y_2, \cdots \rangle$ in G^{-1} such that y_0 is x and for all i, $Order(y_i)$ is not defined. If all the y_i's were distinct, they would constitute an infinite finitary path in G^{-1}, thereby contradicting that G is finitarily anti-founded. Therefore, there exist m and n such that $m < n$ and $y_m = y_n$. Now take $y = y_m$. □

Proposition 12 *If α is any ordinal greater than all orders of vertices in G, then for all $\beta \geq \alpha$, $Y_G(\beta) = Y_G(\alpha)$.*

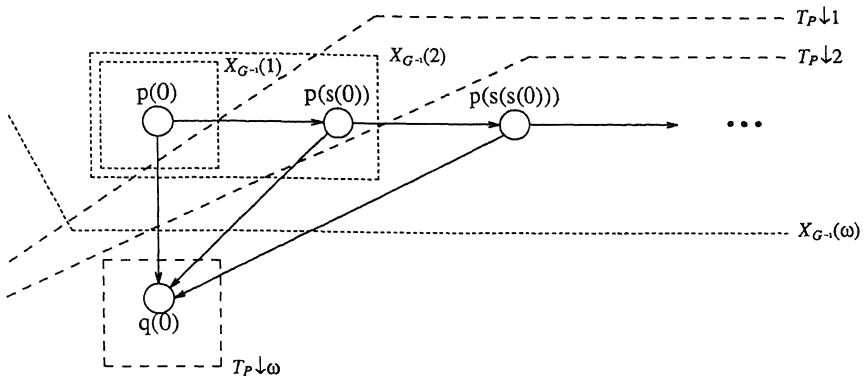

Figure 2: Ordinal powers of T_P compared with Berge's ordinal function

Proof Let α be greater than all orders of vertices in G.

($\subseteq$) Since Y_G is monotonically non-increasing, we have that for all $\beta \geq \alpha$, $Y_G(\beta) \subseteq Y_G(\alpha)$.

($\supseteq$) Let $x \notin Y_G(\beta)$, for some $\beta \geq \alpha$. Then by Proposition 8, there is a maximum ordinal δ such that $x \in Y_G(\delta)$. By definition of order, δ is the order of x. As $\delta < \alpha$, by the assumption on α we have that $x \notin Y_G(\alpha)$. Therefore, $Y_G(\alpha) \subseteq Y_G(\beta)$. □

Theorem 1 *If P is a uniconditional program and G is its associated graph, then for all α, $T_P \downarrow \alpha = Y_G(\alpha) = V \setminus X_{G^{-1}}(\alpha)$.*

Proof By Proposition 4 and Proposition 5. □

See Figure 2 for an example.

Corollary If G is the graph associated with a uniconditional program P, then $\mathrm{dco}(T_P)$ is the least ordinal greater than all orders of vertices in G.

We feel we have now unveiled the secret of some of the examples found in the literature where the downward closure ordinal is greater than ω. Specifically, here is a method to follow if another example is required. Take a directed graph G with at least one vertex of transfinite order, say, α. Name the nodes of G by variable-free atomic formulas. Any uniconditional (possibly infinite) program P of which G is the associated graph then has a downward closure ordinal greater than α. Of course, since logic programs must be finite, this only works properly if P has a finite number of clauses.

Although we have an improvement over the existing situation, where only a few isolated examples of programs with downward closure ordinal exceeding ω were published, the above "method" is hardly satisfactory. It does not specify how to get from an infinite graph G

to a finite, preferably small, logic program having G as associated graph. This problem is addressed in the next section.

4 Graph representations

In this section we consider representations of graphs by logic programs. When G is the graph associated with a uniconditional program P, P is very similar to a graph representation due to Kowalski [7] that we call the *unary representation* of G. According to it, given a graph G whose vertices are labelled by variable-free terms, the clause

$$\mathtt{r}(\sigma) \leftarrow \mathtt{r}(\tau)$$

is in the unary representation P of G if and only if G has an edge directed from the vertex labelled by τ to the vertex labelled by σ. Note that since there is a one-one correspondence between the edges in G and the clauses in P, P can be infinite. The following proposition follows immediately from the corollary to Theorem 1:

Proposition 13 *The downward closure ordinal of the unary representation of a graph is the least ordinal greater than all orders of vertices in the graph.*

Kowalski also uses what we call the *binary representation* of a graph, where a variable-free atom $\mathtt{arc}(\tau, \sigma)$ is interpreted as saying that the graph contains an edge directed from the vertex labelled by τ to the vertex labelled by σ. A binary representation P of a graph G is an axiomatization in Horn clauses of the `arc` relation. The clauses in P can either be all variable-free or, more interestingly, they may contain variables, in which case P can be finite as long as the edge set of G is recursively enumerable. In its general form, P is a binary representation of a graph when

$$\mathtt{arc}(\tau, \sigma) \in T_P \uparrow \omega$$

if and only if the graph contains an edge directed from τ to σ.

Unfortunately, we do not have a result equivalent to Proposition 13 for a binary representation of a graph. This is due to the fact that, unlike the unary representation, a graph may have many binary representations, which have different downward closure ordinals. On one extreme, if the binary representation contains only variable-free unit clauses for the `arc` predicate, its downward closure ordinal is 1; on the other extreme, however, there exist binary representations of the empty graph having a transfinite downward closure ordinal. In order to relate downward closure ordinals of logic programs with the least ordinal greater than all orders of vertices of a graph we find it useful to combine Kowalski's two graph representations. This approach has also been followed by Blair in [6] and by Apt in [1].

A *combined representation* is obtained by adding clauses `C1`,`C2`,`C3` to a binary representation.

```
C1:  r(X) <- r(Y) & arc(Y,X)
```

(assuming that the predicate symbol `r` does not occur in the binary representation). The intuition behind the clause `C1` of the combined representation is that if a vertex Y is reachable and there is an edge from Y to a vertex X, then X is reachable.

Let ARC be a binary representation of a graph G. Note that the edge set of G is determined by the least fixpoint of T_{ARC}, which may not be equal to the greatest fixpoint yielded by the downward closing procedure. This difficulty is overcome by restricting ourselves to determinate binary representations, which have a unique fixpoint yielded by at most ω steps

of the downward closing procedure. This is not too restrictive: due to a result of Blair [5] every recursive relation can be computed by a determinate program. The idea behind C1 is that it transforms a binary representation of a graph into a unary representation. However, even with a determinate program ARC it takes ω steps in the downward closing procedure before arc properly defines the edge set of G. It is clearly undesirable that the unary representation using r is untimely affected during the first ω steps of the closing procedure. Therefore we add to ARC not only C1, but also the following two clauses, which ensure that all variable-free r-atoms 'survive' the initial ω steps of the closing procedure.

```
C2:  r(X) <- p(Y);
C3:  p(s(X)) <- p(X);
```

For technical reasons we require that the predicate symbol p does not occur in ARC, but that the function symbol s does.

Definition Let P be a combined representation of a graph. The clauses with the predicate symbols r or p in their heads (i.e. C1–3) are *kernel* clauses; all other clauses of P are called *non-kernel* and form a logic program denoted by $\hat{P}$.

Definition For any Herbrand interpretation I and predicate symbol p, the p-*component* of I, denoted $I \diamond \mathtt{p}$, is $\{\mathtt{p}(t_1,\ldots,t_n) | \mathtt{p}(t_1,\ldots,t_n) \in I\}$.

Proposition 14 *Let P be a combined representation of a graph. Then*

(a) *for all $\alpha \geq \omega$, $(T_P \downarrow \alpha) \diamond \mathtt{p} = \emptyset$,*

(b) *for all α, $\{T_{\hat{P}} \downarrow \alpha, (T_P \downarrow \alpha) \diamond \mathtt{r}, (T_P \downarrow \alpha) \diamond \mathtt{p}\}$ is a disjoint partition of $T_P \downarrow \alpha$.*

Proof (a) Straightforward as C3 is the only clause in P with the symbol p in its head.

(b) Straightforward since $\hat{P} \subseteq P$, they share the same Herbrand universe U_P and the symbols r and p do not occur in $\hat{P}$. □

Theorem 2 *Let P be a combined representation of a graph $G = \langle V, E\rangle$ such that $\hat{P}$ is determinate. Then*

(a) $(T_P \downarrow \omega) \diamond \mathtt{arc} = \{\mathtt{arc}(\tau,\sigma) | \langle \tau, \sigma\rangle \in E\}$,

(b) $(T_P \downarrow \omega) \diamond \mathtt{r} = \{\mathtt{r}(\sigma) | \sigma \in U_P\} \supseteq \{\mathtt{r}(\sigma) | \sigma \in V\}$,

(c) *for all $\alpha > 0$,* $(T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r} = \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}$.

Proof (a) By proposition 14(b) and the fact that $\hat{P}$ is determinate, we have

$$(T_P \downarrow \omega) \diamond \mathtt{arc} = (T_{\hat{P}} \downarrow \omega) \diamond \mathtt{arc} = (T_{\hat{P}} \uparrow \omega) \diamond \mathtt{arc}.$$

The result follows since $\hat{P}$ is a binary representation of G.

(b) Due to clause C3, for all $n < \omega$, $(T_P \downarrow n) \diamond \mathtt{p} \neq \emptyset$. So by clause C2 we have $(T_P \downarrow \omega) \diamond \mathtt{r} = \{\mathtt{r}(\sigma) | \sigma \in U_P\} \supseteq \{\mathtt{r}(\sigma) | \sigma \in V\}$.

(c) See appendix. □

Theorem 3 *Let P be a combined representation of a non-empty graph G such that $\hat{P}$ is determinate. Then $dco(T_P) = \omega + \alpha$, where α is the least ordinal greater than all orders of vertices in G.*

Proof See appendix. □

4.1 An example of combined representation

Recall that ϵ_0 is the least fixpoint of $\lambda\alpha[\omega^\alpha]$. Consider any graph containing exactly one vertex of order α, for each ordinal $\alpha < \epsilon_0$ and no vertices without orders. It is easily seen that all such graphs have the same transitive closure, denoted by W, which has the property that an edge directed from vertex u to v exists in W iff $Order(u) < Order(v)$. In this section we construct a combined representation of W.

Any combined representation of W will contain the kernel clauses:

```
C1:  r(X) <- r(Y) & arc(Y,X);
C2:  r(X) <- p(Y);
C3:  p(s(X)) <- p(X);
```

along with the non-kernel clauses axiomatizing the arc relation, which depend upon the structure of W.

Since W contains exactly one vertex of each order less than ϵ_0, we can represent vertices by their orders. For representing natural numbers we use variable-free terms made from the constant 0 and the successor function symbol s, i.e. zero is represented by 0, one by s(0), two by s(s(0)) and so on. For any natural number n, we let $\bar{n}$ denote such a term representation of n. To represent ordinals we use the following well-known result of their normal form expansions in base ω (see Sierpinski [10]):

Proposition 15 *Every ordinal number α, such that $0 < \alpha < \epsilon_0$, may be represented uniquely as*

$$\alpha = \omega^{\beta_1}c_1 + \omega^{\beta_2}c_2 + \cdots + \omega^{\beta_n}c_n$$

where n and $c_1, c_2, \ldots, c_n$ are non-zero natural numbers while $\beta_1, \beta_2, \ldots, \beta_n$ is a decreasing sequence of ordinals less than α.

As every ordinal number has a unique normal form, the representation of any ordinal number α, denoted $\lceil\alpha\rceil$, is the list of exponent-coefficient pairs appearing in the same order as in its normal form. Using the function symbol d to construct such pairs, $\lceil\alpha\rceil$ is given by the list

$$\langle \mathtt{d}(\lceil\beta_1\rceil,\bar{c_1}), \mathtt{d}(\lceil\beta_2\rceil,\bar{c_2}), \cdots, \mathtt{d}(\lceil\beta_n\rceil,\bar{c_n})\rangle.$$

By convention we let $\lceil 0\rceil$ be the empty list $\langle\rangle$. Note that a finite number $n > 0$ has different representations as a natural number and as an ordinal, since $\bar{n} = \mathtt{s}^n(0)$ whereas $\lceil n\rceil = \langle \mathtt{d}(\langle\rangle, \mathtt{s}^n(0))\rangle$; also $\bar{0} = 0$ but $\lceil 0\rceil = \langle\rangle$.

As neither addition nor multiplication among ordinals is commutative, arithmetic becomes rather unfamiliar. After some simplification, it can be seen for example that the ordinal

$$(\omega + 1)2(\omega + 1)3(\omega + 1)4$$

has the following normal form

$$\omega^3 4 + \omega^2 3 + \omega 2 + 1$$

and is represented by the list

$$\begin{aligned}\langle \mathtt{d}(\langle \mathtt{d}(\langle\rangle, \mathtt{s}(\mathtt{s}(\mathtt{s}(0))))\rangle, \mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{s}(0))))),\\ \mathtt{d}(\langle \mathtt{d}(\langle\rangle, \mathtt{s}(\mathtt{s}(0)))\rangle, \mathtt{s}(\mathtt{s}(\mathtt{s}(0)))),\\ \mathtt{d}(\langle \mathtt{d}(\langle\rangle, \mathtt{s}(0))\rangle, \mathtt{s}(\mathtt{s}(0))),\\ \mathtt{d}(\langle\rangle, \mathtt{s}(0))\rangle.\end{aligned}$$

We allow lists to be nested to any finite depth. It may be verified that, with one level of nesting this provides a representation for all ordinals up to (but not including) ω; with two levels of nesting, up to ω^ω; and so on. Thus, in the general case, we have a representation for all ordinals smaller than ϵ_0.

W contains an edge from vertex u to vertex v iff $Order(u) < Order(v)$. This gives rise to the following Horn clause C4 for the arc relation:

```
C4:  arc(X,Y) <- ord(X) & ord(Y) & lto(X,Y);
```

The predicate ord is true of all lists that are representations of ordinals. The predicate lto specifies the less-than relation on ordinals: $\texttt{lto}(X,Y)$ is true if the ordinal represented by X is less than that represented by Y. Representing lists as terms made up in the usual way from the constant nil and the binary functor '.', we can axiomatize ord as follows:

```
C5:  ord(nil);
C6:  ord(d(B,s(N)).nil) <- ord(B) & int(N);
C7:  ord(d(B1,s(N1)).d(B2,s(N2)).Rest) <-
                ord(B1) & int(N1) &
                ord(d(B2,s(N2)).Rest) &
                lto(B2,B1);

C8:  int(0);
C9:  int(s(X)) <- int(X);
```

Note that ord requires the coefficient fields to be non-zero and the pairs in the lists to be sorted in decreasing order of their exponent fields.

The $<$ relation on the ordinals induces a $<$ relation on their list representations such that $\lceil\alpha\rceil < \lceil\beta\rceil$ iff $\alpha < \beta$, for any ordinals $\alpha, \beta < \epsilon_0$. We have

$$\langle \mathtt{d}(\lceil\beta_1\rceil,\bar{c_1}),\cdots,\mathtt{d}(\lceil\beta_m\rceil,\bar{c_m})\rangle < \langle \mathtt{d}(\lceil\delta_1\rceil,\bar{d_1}),\cdots,\mathtt{d}(\lceil\delta_n\rceil,\bar{d_n})\rangle$$

if one of the following (mutually exclusive) cases hold:

- $m = 0, n > 0$;
- $m, n > 0$ and $\beta_1 < \delta_1$;
- $m, n > 0$ and $\beta_1 = \delta_1$ and $c_1 < d_1$;
- $m, n > 0$ and $\beta_1 = \delta_1$ and $c_1 = d_1$ and $\langle \mathtt{d}(\lceil\beta_2\rceil,\bar{c_2}),\cdots,\mathtt{d}(\lceil\beta_m\rceil,\bar{c_m})\rangle < \langle \mathtt{d}(\lceil\delta_2\rceil,\bar{d_2}),\cdots,\mathtt{d}(\lceil\delta_n\rceil,\bar{d_n})\rangle$.

These cases are directly translated to the following axioms for the lto predicate:

```
C10: lto(nil,X.Rest);
C11: lto(d(B1,N1).Rest1,d(B2,N2).Rest2) <- lto(B1,B2);
C12: lto(d(B,N1).Rest1,d(B,N2).Rest2) <- ltn(N1,N2);
C13: lto(d(B,N).Rest1,d(B,N).Rest2) <- lto(Rest1,Rest2);
```

The predicate ltn specifies the less-than ordering on natural numbers: $\texttt{ltn}(X,Y)$ is true if the natural number X is less than the natural number Y. It is defined as:

```
C14: ltn(0,s(X));
C15: ltn(s(X),s(Y)) <- ltn(X,Y);
```

This concludes the combined representation of the graph W. Clauses C1-C3 are the kernel clauses and clauses C4-C15 are non-kernel.

5 A family of logic programs

In this section we construct a family $\{P_\alpha\}_{\alpha<\epsilon_0}$ of logic programs, such that $\mathrm{dco}(T_{P_\alpha})$ is $\omega+\alpha$. The members of this family are combined representations of graphs containing vertices with transfinite order.

As a basis for this family we consider the graph W introduced in the previous section, which contains exactly one vertex of each order less than ϵ_0 and has the property that from any vertex there are edges to all vertices with higher orders. Let W_α be the graph obtained by adding, in W, an edge from the vertex with order α to itself. This extra edge introduces a cycle containing only that vertex, thereby causing that vertex, and vertices which in W had a higher order, not to have an order. Thus W_α is, as opposed to W, not acyclic, but W_α is certainly finitarily anti-founded. The program P_α is the combined representation of W_α, containing, in addition to the clauses C1-C15 given in Section 4.1 for the combined representation of W, the non-kernel clause

$$\texttt{C16}: \texttt{arc}(\lceil\alpha\rceil, \lceil\alpha\rceil);$$

Theorem 4 *For $\alpha > 0$, $dco(T_{P_\alpha}) = \omega + \alpha$.*

Proof $\{\texttt{C5}, \ldots, \texttt{C16}\}$ can be verified to meet the conditions of Proposition 3. Thus by Proposition 2, these clauses form a determinate program. Since C4 simply defines the arc relation in terms of ord and lto, it follows that $\hat{P}_\alpha = \{\texttt{C4}, \ldots, \texttt{C16}\}$ is determinate. Proposition 11 tells us that a vertex has an order in W_α iff the corresponding vertex has an order $< \alpha$ in W. So α is the least ordinal greater than all orders of vertices in W_α. The result follows from Theorem 3. □

6 Conclusions

We have presented a systematic way of constructing logic programs with downward closure ordinals up to ϵ_0, the least fixpoint of $\lambda\alpha[\omega^\alpha]$. Given Blair's result that there exist programs with downward closure ordinals up to and including the least non-recursive ordinal (ω_1^{ck}), it is quite tempting to go beyond ϵ_0. Even though we have not considered that in this paper, it seems to be a simple matter to arrive at closer approximations to ω_1^{ck}. Note that our representation of ordinals must be recursive (since the least fixpoint of the T_P function of a determinate program P can easily seen to be recursive), so that our method will never yield ω_1^{ck}. The problem is essentially that of denoting ordinals by logic terms. We have presented a method to construct notations $\lceil\alpha\rceil$ for $\alpha < \epsilon_0$ given some notations $\bar{n}$ for $n < \omega$. This step can be iterated and increasingly large initial segments of ordinals can be assigned notations by enlarging the base of their normal form expansions. For example, instead of ω, using ϵ_0 as the base yields a notation for all ordinals less than the least fixpoint of the function $\lambda\alpha[\epsilon_0^\alpha]$. Rogers [9] gives notation systems to go beyond even this value as follows. Let

$$\begin{aligned} \gamma_0 &= \epsilon_0, \\ \gamma_{n+1} &= \text{least ordinal not expressible as} \\ &\quad \text{a polynomial of } \gamma_n. \end{aligned}$$

Let us call a system of notations *maximal* if it assigns a notation to every recursive ordinal. Then, since for all n, $\gamma_n < \omega_1^{ck}$, any notation system for ordinals up to γ_n will fail to be maximal. A system of notations is said to be *recursively related* if there exists an effective procedure, which, when given any two representations in that system can tell us which

one represents a smaller ordinal. A classical result by Kleene (see Rogers [9]) states that there is no maximal recursively related system of notations. As we need our system to be recursively related (to axiomatize the `lto` predicate), it follows that our method cannot be generalized to a maximal family of programs.

7 Acknowledgements

We would like to thank Krzysztof Apt, Roland Bol and Gary Miller for helpful discussions.

8 Appendix

Proof of Theorem 2(c): By transfinite induction we shall prove for all $\alpha \geq 1$

$$(T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r} \;=\; \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}. \tag{1}$$

By Theorem 2(b) we can only have (1) for $\alpha = 0$ if $V = U_P$.

Basis step:

$$\begin{aligned}
&(T_P \downarrow (\omega + 1)) \diamond \mathtt{r} \\
&\quad = \; (T_P(T_P \downarrow \omega)) \diamond \mathtt{r} \\
&\qquad \text{by definition of } T_P \downarrow (\omega + 1) \\
&\quad = \; \{\mathtt{r}(\sigma) | \exists \tau : \mathtt{r}(\tau) \in T_P \downarrow \omega \text{ and } \mathtt{arc}(\tau, \sigma) \in T_P \downarrow \omega\} \\
&\qquad \text{by the definition of } T_P, \text{ clause } \mathtt{C1} \text{ and Proposition 14(a)} \\
&\quad = \; \{\mathtt{r}(\sigma) | \langle \tau, \sigma \rangle \in E\} \\
&\qquad \text{by Theorem 2(a),(b)} \\
&\quad = \; \{\mathtt{r}(\sigma) | \sigma \in Y_G(1)\}
\end{aligned}$$

Induction step: Assume for some $\alpha > 0$ we have for all $1 \leq \beta < \alpha$

$$(T_P \downarrow (\omega + \beta)) \diamond \mathtt{r} \;=\; \{\mathtt{r}(\sigma) | \sigma \in Y_G(\beta)\}.$$

In proving (1) we distinguish two cases.
If α is a successor ordinal then we have

$$\begin{aligned}
&(T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r} \\
&\quad = \; (T_P(T_P \downarrow (\omega + \alpha - 1))) \diamond \mathtt{r} \\
&\qquad \text{by definition of } T_P \downarrow (\omega + \alpha) \\
&\quad = \; (T_P(T_{\hat{P}} \downarrow (\omega + \alpha - 1) \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r} \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{p})) \diamond \mathtt{r} \\
&\qquad \text{by Proposition 14(b)} \\
&\quad = \; (T_P(T_{\hat{P}} \downarrow (\omega + \alpha - 1) \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r})) \diamond \mathtt{r} \\
&\qquad \text{by Proposition 14(a)} \\
&\quad = \; (T_P(T_{\hat{P}} \downarrow \omega \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r})) \diamond \mathtt{r} \\
&\qquad \text{by Proposition 1 since } \hat{P} \text{ is determinate} \\
&\quad = \; \{\mathtt{r}(\sigma) | \exists \tau : \mathtt{arc}(\tau, \sigma) \in T_{\hat{P}} \downarrow \omega \text{ and } \mathtt{r}(\tau) \in (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r}\} \\
&\qquad \text{by the definition of } T_P, \text{ clause } \mathtt{C1} \text{ and Proposition 14(a)}
\end{aligned}$$

$= \{\mathtt{r}(\sigma) | \exists \tau : \langle \tau, \sigma \rangle \in E \text{ and } \tau \in Y_G(\alpha - 1)\}$
by the induction hypothesis, Proposition 14(b) and Theorem 2(a)
$= \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}$
by definition of Y_G.

If α is a limit ordinal then

$(T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r}$
$= (\bigcap \{T_P \downarrow \beta | \beta < \omega + \alpha\}) \diamond \mathtt{r}$
by definition of $T_P \downarrow (\omega + \alpha)$
$= (\bigcap \{T_P \downarrow (\omega + \beta) | 1 \leq \beta < \alpha\}) \diamond \mathtt{r}$
since $T_P \downarrow$ is decreasing and α is at least ω
$= (\bigcap \{\{\mathtt{r}(\sigma) | \sigma \in Y_G(\beta)\} | 1 \leq \beta < \alpha\}$
by the induction hypothesis
$= (\{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}$
by definition of $Y_G(\alpha)$, since Y_G is decreasing and α is at least ω. □

Proof of Theorem 3: Let α be the least ordinal greater than all orders of vertices in G. We have to prove $\mathrm{dco}(T_P) = \omega + \alpha$. We first prove $\leq$, then $\geq$.

$T_P \downarrow (\omega + \alpha + 1)$
$= T_{\hat{P}} \downarrow (\omega + \alpha + 1) \cup (T_P \downarrow (\omega + \alpha + 1)) \diamond \mathtt{r} \cup (T_P \downarrow (\omega + \alpha + 1)) \diamond \mathtt{p}$
by Proposition 14(b)
$= T_{\hat{P}} \downarrow (\omega + \alpha + 1) \cup (T_P \downarrow (\omega + \alpha + 1)) \diamond \mathtt{r}$
by Proposition 14(a)
$= T_{\hat{P}} \downarrow (\omega + \alpha) \cup (T_P \downarrow (\omega + \alpha + 1)) \diamond \mathtt{r}$
by Proposition 1
$= T_{\hat{P}} \downarrow (\omega + \alpha) \cup \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha + 1)\}$
by Theorem 2(c)
$= T_{\hat{P}} \downarrow (\omega + \alpha) \cup \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}$
by Proposition 12
$= T_{\hat{P}} \downarrow (\omega + \alpha) \cup (T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r}$
by Theorem 2(c)
$= T_P \downarrow (\omega + \alpha)$
by Proposition 14.

Thus, $\mathrm{dco}(T_P) \leq \omega + \alpha$.

By the condition on α, for all $\beta < \alpha$, there is a vertex σ in G such that $\sigma \in Y_G(\beta)$ and $\sigma \notin Y_G(\alpha)$. By Theorem 2(c), $\mathtt{r}(\sigma) \in T_P \downarrow (\omega + \beta)$ but $\mathtt{r}(\sigma) \notin T_P \downarrow (\omega + \alpha)$. Hence, $\mathrm{dco}(T_P) \geq \omega + \alpha$. □

References

[1] K. R. Apt. Introduction to logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, North Holland Publishing Company, Amsterdam, to appear.

[2] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, July 1982.

[3] C. Berge. *The Theory of Graphs.* John Wiley and Sons, 1962.

[4] M. Bezem. Characterizing termination of logic programs with level mappings. In *Proceedings of the North American Conference on Logic Programming*, Cleveland, Ohio, 1989.

[5] H. A. Blair. Decidability in the herbrand base. Manuscript presented at the Workshop on Foundations of Deductive Databases and Logic Programming, Washington, D.C., August 1986.

[6] H. A. Blair. The recursion-theoretic complexity of the semantics of predicate logic as a programming language. *Information and Control*, July–September 1982.

[7] R. Kowalski. *Logic for Problem Solving.* North-Holland, 1979.

[8] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[9] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* MIT Press, 1987.

[10] W. Sierpinski. *Cardinal and Ordinal Numbers.* Polish Scientific Publishers, 1965.

[11] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23, 1976.