Centrum voor Wiskunde en Informatica

Centre for Mathematics and Computer Science

R. Bagai, M. Bezem, M.H. van Emden

On downward closure ordinals of logic programs

Computer Science/Department of Software Technology Report CS-R8917 April

1989

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

R. Bagai, M. Bezem, M.H. van Emden

On downward closure ordinals of logic programs

Computer Science/Department of Software Technology Report CS-R8917 April

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

On Downward Closure Ordinals of Logic Programs

Rajiv Bagai

Dept. of Computer Science, University of Victoria, Victoria B.C., Canada V8W 2Y2

Marc Bezem

Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Maarten van Emden

Dept. of Computer Science, University of Victoria, Victoria B.C., Canada V8W 2Y2

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 omega 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 $\{P_\alpha\}_{\alpha<\omega_1}$ of logic programs where any member P_α has downward closure ordinal $\omega + \alpha$.

Key Words & Phrases: logic program, downward closure ordinal, graph.
1985 Mathematics Subject Classification: 03Dxx, 68Qxx, 68T15
1982 CR Categories: F.3, F.4.1, I.2.3.

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 the mapping. Such fixpoints are subject to an order, so that we can distinguish the greatest and the least fixpoints as being of particular interest. In logic programs, the least fixpoint characterizes terminating behaviour; the difference between the least and the greatest fixpoints can be related to nonterminating behaviour.

The least fixpoint is the 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 the 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

Report CS-R8917
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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 [3] that for every ordinal up to and including the least non-recursive ordinal (ω_1^{ck}), there is a logic program having it as downward closure ordinal. However, given such an ordinal and Blair's proof, it is not 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.

A basis of our approach is provided by the well-known notion of ordering the vertices of an acyclic directed graph by assigning to each vertex an ordinal number, which may or may not be finite. The other basis is a connection between graphs and logic programs 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

The *immediate-consequence function* T_P associated with a logic program P 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 in Lloyd [5], the immediate-consequence function, with respect to the partial order of set inclusion, has a unique least fixpoint (denoted lfp(T_P)) and a unique greatest fixpoint (denoted gfp(T_P)). Approximations to one of the fixpoints are obtained by means of a sequence of sets having the fixpoint as limit.

It is easily shown that T_P is monotonic with respect to set inclusion; thus 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; here B_P is the Herbrand base, the Herbrand interpretation consisting of all variable-free atomic formulas constructible from

symbols in P. The limit of this sequence is its intersection, which is not necessarily a fixpoint of T_P, as the following example shows.

Example 1 Let P be the program

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

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

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

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

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

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

Dually,

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

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

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

As shown in Lloyd [5], both the upward and downward closure ordinals of T_P exist for any logic program P. Moreover, the upward closure ordinal is at most ω (assuming that the right hand sides of clauses in P are finite).

For the program of Example 1, the downward closure ordinal is $\omega + 1$. This shows that for some programs this value can exceed ω. On the other hand, certain classes of programs have downward closure ordinals of at most ω; for example, if in every clause the conclusion contains 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 class is that of determinate programs as defined in Blair [3]:

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

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

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

Determinacy of a program is a fixpoint-theoretic property. We define a stronger proof-theoretic property of programs using the notion of SLD-derivation, which is described in Lloyd [5].

Definition A program is *well-founded* if no infinite SLD-derivation starts from a negative clause consisting of a single variable-free atomic formula.

Proposition 2 *Every well-founded program is determinate.*

Proof Let P be a well-founded program and $A \in B_P$, i.e. A is any variable-free atom. Since any SLD-derivation starting from the negative clause $\leftarrow A$ 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 Lloyd [5]) or in the success set (equal to $T_P \uparrow \omega$). 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

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

However, P is not well-founded since the variable-free clause `<- q` has an infinite derivation, namely

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

For use in later sections we establish here the following result:

Proposition 3 *Let P be a logic program such that all predicate symbols in it 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 well-founded.*

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

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. The example shown above, first published in [1], is due in part to K.L. Clark and in part to H. Andreka and I. Nemeti. Here we discuss the other examples found in Lloyd [5].

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

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

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 [5] 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 not make clear what the *mechanism* is; *why* the examples work. In this section we show that all uniconditional logic programs, that is, those where every clause has one condition, can be mapped to a graph in such a way that each vertex has an ordinal number associated with it having the property that the downward closure ordinal of the program has a simple relationship to the orders of the vertices as defined in graph theory. As it is easier to construct graphs in such a way that all successive ordinals up to a certain transfinite bound are associated with a 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 is the topic of the next two sections.

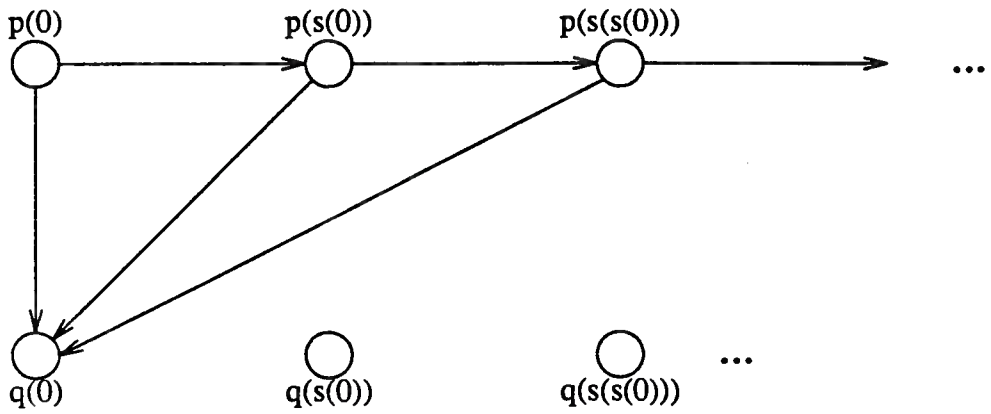

Figure 1: Graph of Example 1

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 one of the examples discussed in the previous section. We will now review known concepts in graph theory that translate 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 $R_G : 2^V \rightarrow 2^V$ 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, 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 was first defined in Berge [2], where it is called "ordinal function". Intuitively, $X_G(\alpha)$ is the set of all vertices v such that all paths in G^{-1} terminating at v are of order type less than α.

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 as follows:

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

In other words, $Y_G(\alpha)$ is the set of all vertices which terminate some path in G of order type α or greater. The following 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 [2] 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 originates. Moreover, if it assigns an order α to a vertex then, for all $\beta < \alpha$ it also assigns order β to some vertex.

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* $\beta < \alpha$ *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 largest 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.

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, x^* is the set of all vertices from which there is a path to x.

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

Definition A graph G is *well-founded* if G^{-1} does not contain any infinite paths.

Proposition 11 *If P is a well-founded uniconditional program having G as its associated graph, then G is well-founded.*

Proposition 12 *For any vertex x of a well-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, $C \cup P \subseteq R_G(C \cup P)$. 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 need to show that there is a $y \in x^*$ such that $y \in y^*$.

Let y_0 be x. By proposition 10, there is a vertex $y_1 \in y_0^*$ such that $Order(y_1)$ is not defined. By iterating the same argument we get an infinite sequence $\langle y_0, y_1, y_2, \cdots \rangle$ of vertices such that y_0 is x and for all i, $Order(y_i)$ is not defined and $y_{i+1} \in y_i^*$. If all the y_i's were distinct, they would constitute an infinite path in G^{-1}, thereby contradicting the well-foundedness of G. Therefore, there exist m and n such that $m < n$ and $y_m = y_n$. Hence, $y_m \in y_m^* \subseteq y_0^*$. □

Proposition 13 *If α is any ordinal greater than all orders of vertices in G, then for all $\beta \geq \alpha$, $Y_G(\beta) = Y_G(\alpha)$.*

Proof ($\subseteq$) 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, $T_P \downarrow \alpha = Y_G(\alpha) = V \setminus X_{G^{-1}}(\alpha)$.*

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 any acyclic graph G with at least one vertex of transfinite order, say, α. Name the nodes of G by variable-free atomic formulas. A uniconditional (possibly infinite) program P of which G is the associated graph then has a downward closure ordinal greater than α. Of course we choose G and name the nodes in such a way that P has a finite, even a small 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.

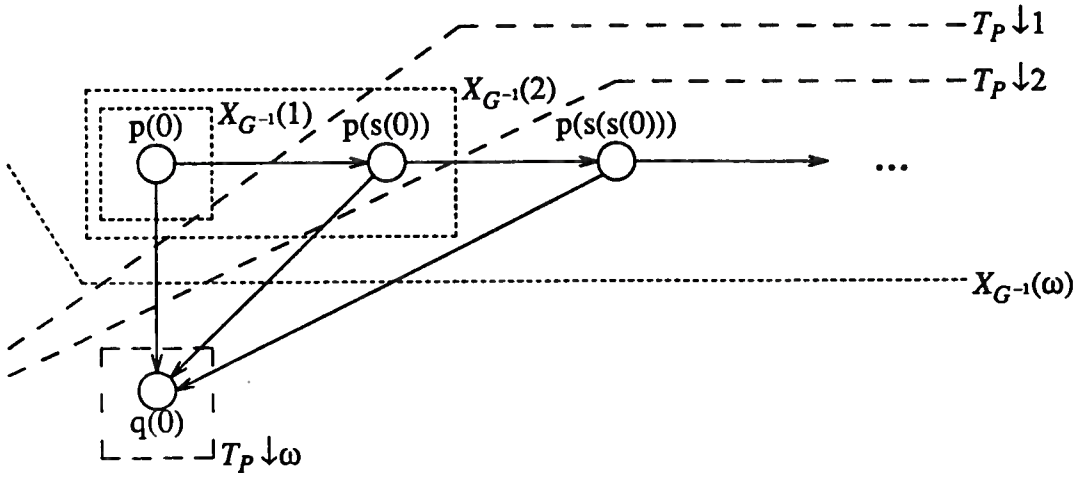

Figure 2: Ordinal powers compared with Berge's ordinal function

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 similar to a graph representation due to Kowalski [4] that we call *unary representation*. 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 P iff 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 greatest fixpoint for the unary representation is easily seen to be the empty set and the following proposition follows immediately from the corollary to theorem 1:

Proposition 14 *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 if the edge set of G is recursive. In its general form, P is a binary representation of a graph when $\mathtt{arc}(\tau, \sigma) \in T_P \uparrow \omega$ iff the graph contains an edge directed from τ to σ.

Unfortunately, we do not have a result equivalent to proposition 14 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; however on the other extreme, there may exist binary representations with higher, even transfinite, downward closure ordinals.

We find it useful to combine Kowalski's two graph representations. A *combined representation* is obtained by adding the following clauses to a binary representation:

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

assuming that the predicate symbols r and p do not occur in the binary representation but the function symbol s does. The intuition behind the clause C1 of the combined representation is that if a vertex Y is reachable and there is an arc from Y to a vertex X, then X is reachable.

The reason for including clauses C2 and C3 in a combined representation becomes clear in the proof of theorem 2(b), but for an intuitive understanding, first consider a combined representation, say R, without these two clauses. Assuming that the corresponding binary representation contains only variable-free unit clauses defining the arc relation, from proposition 14 it may be seen that $\mathrm{dco}(T_R)$ will be $1 + \alpha$, where α is the least ordinal greater than all orders of vertices in the graph. In general, if the downward closure ordinal of the binary representation is β, $\mathrm{dco}(T_R)$ will be at most $\beta + \alpha$; it will not always be equal to $\beta + \alpha$ because the r-atoms may start disappearing in the sequence $\langle T_R \downarrow \delta \rangle_{\delta \geq 0}$ before all the unwanted arc-atoms have disappeared. To ensure that it is equal to $\beta + \alpha$ we need to retain all the r-atoms just until all the unwanted arc-atoms have disappeared. For this reason, we will be particularly interested in determinate binary representations, so that all the unwanted arc-atoms disappear within the first ω steps of the sequence $\langle T_R \downarrow \delta \rangle_{\delta \geq 0}$. Clauses C2 and C3 are added in R to retain all the r-atoms until $T_R \downarrow \omega$. The downward closure ordinal of T_R then turns out to be $\omega + \alpha$.

Definition Let P be a combined representation of a graph. The clauses with the predicate symbols r or p in their heads are *kernel* clauses; all other clauses of P are called *non-kernel.*

Definition If P is a combined representation of a graph, $\hat{P}$ denotes the set of all non-kernel clauses of 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 15 *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) $(T_P \downarrow \omega) \diamond \mathtt{r} \supseteq \{\mathtt{r}(\sigma) | \sigma \in V\}$,

(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 15(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 $n < \omega$, $(T_P \downarrow n) \diamond \mathtt{p} \neq \emptyset$. The result follows due to clause C2.[1]

(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. □

[1] In fact, clauses C2 and C3 are included in the combined representation only to ensure that the set $\{\mathtt{r}(\sigma) | \sigma \in V\}$ is contained in $T_P \downarrow \omega$. The importance of this becomes clear in the proof of part (c).

4.1 An example of combined representation

Consider any graph containing exactly one vertex of order α, for each ordinal $\alpha < \epsilon_0$. 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[2] we use the following well-known result of their normal form expansions in base ω (see Sierpinski [7]):

Proposition 16 *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.

When the β's are finite, α can be any ordinal less than ω^ω. More generally, if the β's are less than ${}^n\omega$, α can be any ordinal less than ${}^{1+n}\omega$. Here, ${}^n\omega$ is a "tower" of n ω's defined by ${}^1\omega = \omega$ and ${}^{1+n}\omega = \omega^{({}^n\omega)}$, where n is a natural number. 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 $\bar{n} = \mathtt{s}^n(\mathtt{0})$ whereas $\lceil n \rceil = \langle \mathtt{d}(\langle\rangle, \mathtt{s}^n(\mathtt{0})) \rangle$; also $\bar{0} = \mathtt{0}$ 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

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

[2] That is, there will be two representations for finite ordinals.

has the following normal form

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

and is represented by the list

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

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. With a level of nesting of n we can represent all ordinals less than ${}^n\omega$, the tower of n ω's defined just after proposition 16. Thus, in the general case, we have a representation for all ordinals smaller than ϵ_0, which is the least fixpoint of the function $\lambda\alpha[\omega^\alpha]$.

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: `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(p(B,s(N)).nil) <- ord(B) & int(N);
C7:  ord(p(B1,s(N1)).p(B2,s(N2)).Rest) <-
                ord(B1) & int(N1) &
                ord(p(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(p(B1,N1).Rest1,p(B2,N2).Rest2) <- lto(B1,B2);
C12: lto(p(B,N1).Rest1,p(B,N2).Rest2) <- ltn(N1,N2);
C13: lto(p(B,N).Rest1,p(B,N).Rest2) <- lto(Rest1,Rest2);
```

The predicate `ltn` specifies the less-than ordering on natural numbers: `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. Then the program P_α is the combined representation of W_α.

Thus in addition to the clauses `C1-C15` given in section 4.1 for the combined representation of W, P_α contains the non-kernel clause

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

Theorem 4 *For $\alpha > 0$, $dco(T_P) = \omega + \alpha$.*

Proof $\hat{P}_\alpha$ (= {C4,...,C16}) can be verified to meet the conditions of proposition 3. Thus by proposition 2, it is determinate. Proposition 12 tells us that a vertex has an order in W_α iff the corresponding vertex has an order less than α 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}. 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 [6] 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 classic result by Kleene (see Rogers [6]) is 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) By theorem 2(b) we have

$$\begin{aligned}(T_P \downarrow \omega) \diamond \mathtt{r} &\supseteq \{\mathtt{r}(\sigma) | \sigma \in V\} \\ &= \{\mathtt{r}(\sigma) | \sigma \in Y_G(0)\}.\end{aligned}$$

Induction hypothesis: For all $\beta < \alpha$,

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

Induction step: If α is a successor ordinal then

$$\begin{aligned}&(T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r} \\ &\quad = (T_P(T_P \downarrow (\omega + \alpha - 1))) \diamond \mathtt{r} \\ &\qquad \text{by definition of } T_P \downarrow (\omega + \alpha) \\ &\quad = (T_P(T_{\hat{P}} \downarrow (\omega + \alpha - 1) \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r} \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{p})) \diamond \mathtt{r} \\ &\qquad \text{by proposition 15(b)} \\ &\quad = (T_P(T_{\hat{P}} \downarrow (\omega + \alpha - 1) \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r})) \diamond \mathtt{r} \\ &\qquad \text{by proposition 15(a)} \\ &\quad = (T_P(T_{\hat{P}} \downarrow \omega \cup (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r})) \diamond \mathtt{r} \\ &\qquad \text{by proposition 1} \\ &\quad = \{\mathtt{r}(\sigma) | \exists \tau : \mathtt{arc}(\tau, \sigma) \in T_{\hat{P}} \downarrow \omega \text{ and } \mathtt{r}(\tau) \in (T_P \downarrow (\omega + \alpha - 1)) \diamond \mathtt{r}\} \\ &\qquad \text{by definition of } T_P \text{ on clause } \mathtt{C1}, \text{ and proposition 2} \\ &\quad = \{\mathtt{r}(\sigma) | \exists \tau : \mathtt{arc}(\tau, \sigma) \in T_{\hat{P}} \downarrow \omega \text{ and } \tau \in Y_G(\alpha - 1)\} \\ &\qquad \text{by induction hypothesis and } \tau \in V \\ &\quad = \{\mathtt{r}(\sigma) | \exists \tau : \langle \tau, \sigma \rangle \in E \text{ and } \tau \in Y_G(\alpha - 1)\} \\ &\qquad \text{by proposition 15(b) and theorem 2(a)} \\ &\quad = \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\} \\ &\qquad \text{by definition of } Y_G.\end{aligned}$$

If α is a limit ordinal then

$$\begin{aligned}&(T_P \downarrow (\omega + \alpha)) \diamond \mathtt{r} \\ &\quad = (\bigcap \{T_P \downarrow \beta | \beta < \omega + \alpha\}) \diamond \mathtt{r} \\ &\qquad \text{by definition of } T_P \downarrow (\omega + \alpha) \\ &\quad = (T_P \downarrow \omega \cap \bigcap \{T_P \downarrow (\omega + \beta) | \beta < \alpha\}) \diamond \mathtt{r} \\ &\qquad \text{by simplification} \\ &\quad = (T_P \downarrow \omega) \diamond \mathtt{r} \cap \bigcap \{(T_P \downarrow (\omega + \beta)) \diamond \mathtt{r} | \beta < \alpha\}\end{aligned}$$

by further simplification

$= (T_P \downarrow \omega) \diamond \mathtt{r} \cap \bigcap \{\{\mathtt{r}(\sigma) | \sigma \in Y_G(\beta)\} | \beta < \alpha\}$

by induction hypothesis and induction step for successors

$= (T_P \downarrow \omega) \diamond \mathtt{r} \cap \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}$

by definition of $Y_G(\alpha)$

$= \{\mathtt{r}(\sigma) | \sigma \in Y_G(\alpha)\}$

by theorem 2(b).□

Proof of theorem 3: ($\leq$) For $\beta \geq \alpha$ we have

$T_P \downarrow (\omega + \beta + 1)$

$= T_{\hat{P}} \downarrow (\omega + \beta + 1) \cup (T_P \downarrow (\omega + \beta + 1)) \diamond \mathtt{r} \cup (T_P \downarrow (\omega + \beta + 1)) \diamond \mathtt{p}$

by proposition 15(b)

$= T_{\hat{P}} \downarrow (\omega + \beta + 1) \cup (T_P \downarrow (\omega + \beta + 1)) \diamond \mathtt{r}$

by proposition 15(a)

$= T_{\hat{P}} \downarrow (\omega + \beta) \cup (T_P \downarrow (\omega + \beta + 1)) \diamond \mathtt{r}$

by proposition 1

$= T_{\hat{P}} \downarrow (\omega + \beta) \cup \{\mathtt{r}(\sigma) | \sigma \in Y_G(\beta + 1)\}$

by theorem 2(c)

$= T_{\hat{P}} \downarrow (\omega + \beta) \cup \{\mathtt{r}(\sigma) | \sigma \in Y_G(\beta)\}$

by proposition 13

$= T_{\hat{P}} \downarrow (\omega + \beta) \cup (T_P \downarrow (\omega + \beta)) \diamond \mathtt{r}$

by theorem 2(c)

$= T_P \downarrow (\omega + \beta)$

by proposition 15.

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

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

References

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

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

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

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

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

[6] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.

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