# Chapter 8

# Termination of Programs

# Outline

- Level mappings
- Generally terminating programs: Recurrent programs
- Left terminating programs: Acceptable programs

# Does this Program Terminate?

```
wine(riesling, chicken).
wine(riesling, veal).
wine(kerner, veal).

diff(riesling, kerner).
diff(kerner, riesling).

interchangeable(X, Y) :- wine(X, Z), wine(Y, Z), diff(X, Y).
```

# Do these two Terminate?

```
edge(a, b).
edge(b, c).
edge(d, e).
path(X, Y) :- edge(X, Y).
path(X, Y) :- edge(X, Z), path(Z, Y).
```

```
arc(a, b).
arc(b, c).
arc(d, e).
connected(X, Y) :- arc(X, Y).
connected(X, Y) :- connected(X, Z), arc(Z, Y).
```

# And this one?

```
edge(a, b).
edge(b, c).
edge(d, e).
edge(c, a).

path(X, Y) :- edge(X, Y).
path(X, Y) :- edge(X, Z), path(Z, Y).
```

# What About this one?

```
edge(a, b).
edge(b, c).
edge(d, e).
edge(c, a).

dpath(X, Y, _) :- edge(X, Y).
dpath(X, Y, Depth) :-
      Depth > 0,
      edge(X, Z),
      Depth1 is Depth – 1,
      dpath(Z, Y, Depth1).

path(X, Y) :- dpath(X, Y, 10).
```

# A Difficult one ...

```
jump(1).

jump(N) :-
     N > 1, N mod 2 =:= 1, N1 is 3*N + 1, jump(N1).

jump(N) :-
     N > 1, N mod 2 =:= 0, N1 is N // 2, jump(N1).
```

# Termination May Depend on the Query

```
app([], X, X).
app([X|Y], Z, [X|U]) :- app(Y, Z, U).
```

The query `app([a,b], Y, Z)` terminates.

The query `app(X, Y, [c,d])` terminates.

The query `app(X, [e,f], Z)` does not terminate.

How can we prove that certain programs and queries terminate?

# General vs. PROLOG Termination

```
app([], X, X).
app([X|Y], Z, [X|U]) :- app(Y, Z, U).

app3(X, Y, Z, U) :- app(X, Y, V), app(V, Z, U).
```

Query `app3([a], [b], [c], U)` has an infinite SLD-derivation.

However, PROLOG terminates.

# Multisets

multiset (written $bag(a_1, ..., a_n)$)

:⇔

unordered sequence $a_1, ..., a_n$

$\prec$ (on finite multisets of natural numbers)

:⇔

$X \prec Y$ iff $X = (Y - bag(a)) \cup Z$

for some $a \in Y$ and $Z$ such that $\forall b \in Z.\ b < a$

We write $old(X, Y)$ :⇔ $a$ and $new(X, Y)$ :⇔ $Z$.

Note: $\prec$ is irreflexive and antisymmetric

# Multiset Ordering

transitive closure of a relation $R$ on a set $\mathcal{A}$

$:\Leftrightarrow$

smallest transitive relation on $\mathcal{A}$ that contains $R$

multiset ordering ($\prec_m$) $:\Leftrightarrow$ transitive closure of $\prec$

Theorem 6.4

The multiset ordering $\prec_m$ is well-founded.

# Two Helpful Observations

**Lemma 6.2**

An infinite, finitely branching tree has an infinite branch.

**Note 6.3**

An irreflexive, antisymmetric relation is well-founded iff its transitive closure is well-founded.

Thus finiteness of an SLD-tree (hence, termination) can be proved by finding a suitable multiset assignment for queries.

# Level Mappings

level mapping for program $P$ :⇔ function $|\ | : HB_P \mapsto \mathbb{N}$

level of ground atom $A$ :⇔ $|A|$

clause $c$ recurrent w.r.t. $|\ |$

:⇔

for every ground instance $A \leftarrow \underline{\underline{B}}$ of $c$ and every $B \in \underline{\underline{B}}$:

$|A| > |B|$

program $P$ recurrent :⇔ for some level mapping $|\ |$,

each $c \in P$ is recurrent w.r.t. $|\ |$

# Example (I)

$$member(x, [x|y]) \leftarrow$$
$$member(x, [y|z]) \leftarrow member(x, z)$$

With $|\, member(s, t) \,| :\Leftrightarrow$ "listsize" of $t$, the clauses are recurrent.

$$subset([x|y], z) \leftarrow member(x, z), subset(y, z)$$
$$subset([\,], x) \leftarrow$$

Define $|\, subset(s, t) \,| :\Leftrightarrow listsize(s) + listsize(t)$.

This shows that the entire program is recurrent.
Incidentally, the program always terminates for ground queries.

# Example (II)

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$rev([\,], [\,]) \leftarrow$
$rev([x|y], z) \leftarrow rev(y, u), app(u, [x], z)$

This program is not recurrent.

Incidentally, it does not always terminate for ground queries.

$$\underline{rev([a, b], c)} \Longrightarrow rev([b], u_1), app(u_1, [a], c)$$
$$\Longrightarrow rev([\,], u_2), app(u_2, [b], u_1), app(u_1, [a], c)$$
$$\Longrightarrow rev([\,], u_2), app(y_3, [b], u_3), app(u_1, [a], c)$$
$$\Longrightarrow \ldots$$

# Bounded Queries

atom $A$ bounded w.r.t. $| \, |$

$:\Leftrightarrow$ for some $k \in \mathbb{N}$ we have $|A'| \leq k$ for all $A' \in \mathit{ground}(A)$

level $|A|$ of bounded atom $A$ $:\Leftrightarrow$ $\mathit{max}\{|A'| \mid A' \in \mathit{ground}(A)\}$

query bounded w.r.t. $| \, |$ $:\Leftrightarrow$ all its atoms are bounded w.r.t. $| \, |$

query $A_1, \ldots, A_n$ bounded by $k$ $:\Leftrightarrow$ $|A_i| \leq k$ for $i = 1, \ldots, n$

level $|Q|$ of bounded query $Q = A_1, \ldots, A_n$

$:\Leftrightarrow$ $\mathit{bag}(|A_1|, \ldots, |A_n|)$

# Boundedness Lemma for Recurrent Programs

Lemma 6.8

Let $P$ be a recurrent (w.r.t. $|\ |$) program. If $Q_1$ is a query bounded w.r.t. $|\ |$ and $Q_2$ an SLD-resolvent of $Q_1$, then

- $Q_2$ is bounded w.r.t. $|\ |$
- $|Q_2| \prec_m |Q_1|$

Proof:

1. Any instance $Q'$ of $Q$ is bounded and satisfies $|Q'| \preceq_m |Q|$.
2. An instance of a recurrent clause is recurrent.
3. For every recurrent $H \leftarrow \underline{\underline{B}}$ and every bounded $\underline{\underline{A}}, H, \underline{\underline{C}}$, $\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}}$ is bounded and satisfies $|\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}}| \prec_m |\underline{\underline{A}}, H, \underline{\underline{C}}|$.

# Finiteness for Recurrent Programs

Corollary 6.9

Let $P$ be a recurrent program and $Q$ a bounded query.
Then all SLD-derivations of $P \cup \{Q\}$ are finite.

# Verifying Termination

listsize of a term $t$ ($|t|$)

$:\Leftrightarrow$

$|[s|t]| = |t| + 1$

$|f(t_1, ..., t_n)| = 0$ if $f \neq [\cdot|\cdot]$

$list([\ ]) \leftarrow$
$list([x|y]) \leftarrow list(y)$

Defining $|list(t)| :\Leftrightarrow |t|$
shows that this program is recurrent,
hence always terminating for bounded queries.

# Importance of Choice of Level Mapping

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$

These clauses are recurrent w.r.t. $|app(x, y, z)|_1 :\Leftrightarrow |x|$

and also w.r.t. $|app(x, y, z)|_2 :\Leftrightarrow |z|$.

In each case we obtain different bounded queries.

E.g., $app([a, b], y, z)$ is bounded w.r.t. $|\ |_1$ but not w.r.t. $|\ |_2$

$app(x, y, [c, d])$ is bounded w.r.t. $|\ |_2$ but not w.r.t. $|\ |_1$

Both these queries are bounded w.r.t.

$$|app(x, y, z)|_3 :\Leftrightarrow min(|x|, |z|)$$

# Limitations: General SLD vs. Prolog (I)

```
edge(a, b).
edge(b, c).
edge(d, e).
path(X, Y) :- edge(X, Y).
path(X, Y) :- edge(X, Z), path(Z, Y).

arc(a, b).
arc(b, c).
arc(d, e).
connected(X, Y) :- arc(X, Y).
connected(X, Y) :- connected(X, Z), arc(Z, Y).
```

Neither program is recurrent.

However, all LD-derivations for the first program are finite.

# Limitations: General SLD vs. Prolog (II)

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$app3(x, y, z, u) \leftarrow app(x, y, v), app(v, z, u)$

$$|app(x, y, z)| \;:\Leftrightarrow\; min(|x|, |z|)$$
$$|app3(x, y, z, u)| \;:\Leftrightarrow\; |x| + |u| + 1$$

shows that the program is recurrent.

But $app3([a], [b], [c], u)$ is not bounded w.r.t. $|\ |$ and indeed has an infinite derivation.

However, all LD-derivations of $P \cup \{app3([a], [b], [c], u)\}$ are finite.

# Acceptable Programs

clause $c$ acceptable w.r.t. level mapping $|\ |$ and interpretation $I$

$:\Leftrightarrow$

$I$ model of $c$,

for every ground instance $A \leftarrow \underline{\underline{A}}, B, \underline{\underline{B}}$ of $c$ and every $B$ such that $I \models \underline{\underline{A}}$:

$$|A| > |B|$$

program $P$ acceptable

$:\Leftrightarrow$ for some level mapping $|\ |$ and interpretation $I$, each $c \in P$ is acceptable w.r.t. $|\ |$ and $I$

# Example (I)

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$rev([\,], [\,]) \leftarrow$
$rev([x|y], z) \leftarrow rev(y, u), app(u, [x], z)$

$$|app(x, y, z)| :\Leftrightarrow min(|x|, |z|)$$
$$|rev(x, y)| :\Leftrightarrow |x|$$
$$I :\Leftrightarrow \{app(x, y, z) \mid |x| + |y| = |z|\} \cup \{rev(x, y) \mid |x| = |y|\}$$

shows that the program is acceptable.

# Example (II)

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$app3(x, y, z, u) \leftarrow app(x, y, v), app(v, z, u)$

$$|app(x, y, z)| :\Leftrightarrow |x|$$
$$|app3(x, y, z, u)| :\Leftrightarrow |x| + |y| + 1$$
$$I :\Leftrightarrow \{app(x, y, z) \mid |x| + |y| = |z|\} \cup ground(app3(x, y, z, u))$$

shows that the program is acceptable.

# Acceptability vs. Recurrence

Note 6.21

A program is recurrent w.r.t. $|\ |$ iff it is acceptable w.r.t. $|\ |$ and $HB$.

# An Extended Notion of Boundedness (I)

Let $|\ |$ be a level mapping, $I$ an interpretation, $k \in \mathbb{N}$.

query $Q$ bounded by $k$ w.r.t. $|\ |$ and $I$

$:\Leftrightarrow$

for every ground instance $\underline{\underline{A}}, B, \underline{\underline{B}}$ of $Q$ such that $I \models \underline{\underline{A}}$,

$$|B| \leq k$$

query $Q$ bounded w.r.t. $|\ |$ and $I$

$:\Leftrightarrow$ $Q$ bounded by some $k$ w.r.t. $|\ |$ and $I$

# Example

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$app3(x, y, z, u) \leftarrow app(x, y, v), app(v, z, u)$

$$|app(x, y, z)| :\Leftrightarrow |x|$$
$$|app3(x, y, z, u)| :\Leftrightarrow |x| + |y| + 1$$
$$I :\Leftrightarrow \{app(x, y, z) \mid |x| + |y| = |z|\} \cup ground(app3(x, y, z, u))$$

The program is acceptable (w.r.t. $|\ |$ and $I$),
and $app3([a], [b], [c], u)$ is bounded (by $k = 3$) w.r.t. $|\ |$ and $I$.

# A Notational Convention

max: $\mathcal{P}(\mathbb{N}) \mapsto \mathbb{N} \cup \{\omega\}$ with

max $S$: $\Leftrightarrow \begin{cases} 0 & \text{if } S = \emptyset \\ n & \text{if } S \text{ is finite but not empty and with maximum } n \\ \omega & \text{if } S \text{ is infinite} \end{cases}$

# An Extended Notion of Boundedness (II)

Let $Q$ be a query consisting of $n \geq 1$ atoms.

Then for every $i = 1, ..., n$ and every interpretation $I$,

$$|Q|_i^I :\Leftrightarrow \{|A_i| : A_1, ..., A_n \text{ ground instance of } Q \\ I \models A_1, ..., A_{i-1}\}$$

If $Q$ is bounded w.r.t. some $|\ |$ and $I$, then

$$|Q|_I :\Leftrightarrow bag(\max |Q|_1^I, ..., \max |Q|_n^I)$$

# Example

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$app3(x, y, z, u) \leftarrow app(x, y, v), app(v, z, u)$

$$|app(x, y, z)| :\Leftrightarrow |x|$$
$$|app3(x, y, z, u)| :\Leftrightarrow |x| + |y| + 1$$
$$I :\Leftrightarrow \{app(x, y, z) \mid |x| + |y| = |z|\} \cup ground(app3(x, y, z, u))$$

$$|app3([a], [b], [c], u)|_I = bag(3)$$
$$|app([a], [b], v_1), app(v_1, [c], u)|_I = bag(1, 2)$$

# Boundedness Lemma for Acceptable Programs

Lemma 6.23

Let $P$ be an acceptable (w.r.t. $|\ |$ and $I$) program. If $Q_1$ is a query bounded w.r.t. $|\ |$ and $I$, and if $Q_2$ is an LD-resolvent of $Q_1$, then

- $Q_2$ is bounded w.r.t. $|\ |$ and $I$
- $|Q_2|_I \prec_m |Q_1|_I$

Proof:

1. Any instance $Q'$ of $Q$ is bounded and satisfies $|Q'|_I \preceq_m |Q|_I$.
2. An instance of an acceptable clause is acceptable.
3. For every acceptable $A \leftarrow \underline{\underline{B}}$ and every bounded $A, \underline{\underline{C}}$, $\underline{\underline{B}}, \underline{\underline{C}}$ is bounded and satisfies $|\underline{\underline{B}}, \underline{\underline{C}}|_I \prec_m |A, \underline{\underline{C}}|_I$.

   (See the book on page 161.)

# Finiteness for Acceptable Programs

Corollary 6.24

Let $P$ be an acceptable program and $Q$ a bounded query.
Then all LD-derivations of $P \cup \{Q\}$ are finite.

# Application

$app([\,], x, x) \leftarrow$
$app([x|y], z, [x|u]) \leftarrow app(y, z, u)$
$perm([\,], [\,]) \leftarrow$
$perm(x, [y|z]) \leftarrow app(u, [y|v], x), app(u, v, w), perm(w, z)$

$$|app(x, y, z)| :\Leftrightarrow min(|x|, |z|)$$
$$|perm(x, y)| :\Leftrightarrow |x| + 1$$
$$I :\Leftrightarrow \{app(x, y, z) \mid |x| + |y| = |z|\} \cup ground(perm(x, y))$$

This shows that the program is acceptable.

# Objectives

- Level mappings
- Generally terminating programs: Recurrent programs
- Left terminating programs: Acceptable programs