Chapter 8

Termination of Programs
Outline

- Level mappings
- Generally terminating programs: Recurrent programs
- Left terminating programs: Acceptable programs
Does this Program Terminate?

\[
\text{wine(riesling, chicken).} \\
\text{wine(riesling, veal).} \\
\text{wine(kerner, veal).} \\

\text{diff(riesling, kerner).} \\
\text{diff(kerner, riesling).} \\

\text{interchangeable(X, Y) :- wine(X, Z), wine(Y, Z), diff(X, Y).}
\]
Do these two Terminate?

definitions:

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

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

definitions:

arc(a, b).
arcc(b, c).
arcc(d, e).

connected(X, Y) :- arc(X, Y).
connected(X, Y) :- connected(X, Z), arc(Z, Y).

Foundations of Logic Programming
Termination of Programs
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).
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 ...

\[
\text{jump}(1).
\]

\[
\text{jump}(N) :- \\
N > 1, \ N \mod 2 =:= 1, \ N1 \text{ is } 3*N + 1, \ \text{jump}(N1).
\]

\[
\text{jump}(N) :- \\
N > 1, \ N \mod 2 =:= 0, \ N1 \text{ is } N \div 2, \ \text{jump}(N1).
\]
Termination May Depend on the Query

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

The query \( \text{app}([a,b], Y, Z) \) terminates.
The query \( \text{app}(X, Y, [c,d]) \) terminates.
The query \( \text{app}(X, [e,f], Z) \) 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 \textit{bag}(a_1, ..., a_n))

\[ \Leftrightarrow \]

unordered sequence \(a_1, ..., a_n\)

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

\[ \Leftrightarrow \]

\(X \prec Y\) iff \(X = (Y - \text{bag}(a)) \cup Z\)

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

We write \textit{old}(X, Y) \(\Leftrightarrow a\) and \textit{new}(X, Y) \(\Leftrightarrow Z\).

\textbf{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 $P : \leftrightarrow \text{function } | | : HB_P \mapsto \mathbb{N}$

level of ground atom $A : \leftrightarrow |A|$

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

$\iff$

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

$|A| > |B|$

program $P$ recurrent $: \leftrightarrow \text{for some level mapping } | |,$

each $c \in P$ is recurrent w.r.t. $| |$
Example (I)

\[
\begin{align*}
\text{member}(x, [x|y]) & \leftarrow \\
\text{member}(x, [y|z]) & \leftarrow \text{member}(x, z)
\end{align*}
\]

With \( \text{member}(s, t) \) \iff “listsize” of \( t \), the clauses are recurrent.

\[
\begin{align*}
\text{subset}([x|y], z) & \leftarrow \text{member}(x, z), \text{subset}(y, z) \\
\text{subset}([], x) & \leftarrow
\end{align*}
\]

Define \( \text{subset}(s, t) \) \iff listsize\( (s) + \text{listsize}(t) \).

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

\[
\begin{align*}
\text{app}([\ ], x, x) & \leftarrow \\
\text{app}([x|y], z, [x|u]) & \leftarrow \text{app}(y, z, u) \\
\text{rev}([\ ], [\ ]) & \leftarrow \\
\text{rev}([x|y], z) & \leftarrow \text{rev}(y, u), \text{app}(u, [x], z)
\end{align*}
\]

This program is not recurrent.
Incidentally, it does not always terminate for ground queries.

\[
\text{rev}([\ a, b, c]) \Rightarrow \text{rev}([\ b], u_1), \text{app}(u_1, [\ a], c) \\
\Rightarrow \text{rev}([\ ], u_2), \text{app}(u_2, [\ b], u_1), \text{app}(u_1, [\ a], c) \\
\Rightarrow \text{rev}([\ ], u_2), \text{app}(y_3, [\ b], u_3), \text{app}(u_1, [\ a], c) \\
\Rightarrow \ldots
\]
Bounded Queries

atom $A$ bounded w.r.t. $| |$ :\[\iff \text{for some } k \in \mathbb{N} \text{ we have } |A'| \leq k \text{ for all } A' \in \text{ground}(A)\]

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

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

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

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

$\iff \text{bag}(|A_1|, \ldots, |A_n|)$
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. $|$
- $|Q_2| \preceq_m |Q_1|$

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 $H \leftarrow B$ and every bounded $A, H, C$,
   $A, B, C$ is bounded and satisfies $|A, B, C| \preceq_m |A, H, C|$.
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

\textbf{listsize} of a term \( t (|t|) \)

\[ \Leftrightarrow \]

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

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

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{list}([ ]) \leftarrow \\
\textbf{list}([x|y]) \leftarrow \textbf{list}(y) \\
\hline
\end{tabular}
\end{center}

Defining \(|list(t)| \Leftrightarrow |t|\)

shows that this program is recurrent,

hence always terminating for bounded queries.
Importance of Choice of Level Mapping

\[
\text{app}([ \ ], x, x) \leftarrow \\
\text{app}([x|y], z, [x|u]) \leftarrow \text{app}(y, z, u)
\]

These clauses are recurrent w.r.t. \(|\text{app}(x, y, z)|_1 : \Leftrightarrow |x|
and also w.r.t. \(|\text{app}(x, y, z)|_2 : \Leftrightarrow |z|.

In each case we obtain different bounded queries.

E.g., \(\text{app}([a, b], y, z)\) is bounded w.r.t. \(|\ |_1\) but not w.r.t. \(|\ |_2\)
\(\text{app}(x, y, [c, d])\) is bounded w.r.t. \(|\ |_2\) but not w.r.t. \(|\ |_1\)

Both these queries are bounded w.r.t.
\(|\text{app}(x, y, z)|_3 : \Leftrightarrow \text{min}(|x|, |z|)\)
Limitations: General SLD vs. Prolog (I)

\[
\begin{align*}
\text{edge}(a, b). \\
\text{edge}(b, c). \\
\text{edge}(d, e). \\
\text{path}(X, Y) & :\text{edge}(X, Y). \\
\text{path}(X, Y) & :\text{edge}(X, Z), \text{path}(Z, Y).
\end{align*}
\]

\[
\begin{align*}
\text{arc}(a, b). \\
\text{arc}(b, c). \\
\text{arc}(d, e). \\
\text{connected}(X, Y) & :\text{arc}(X, Y). \\
\text{connected}(X, Y) & :\text{connected}(X, Z), \text{arc}(Z, Y).
\end{align*}
\]

Neither program is recurrent.
However, all LD-derivations for the first program are finite.
Limitations: General SLD vs. Prolog (II)

\[
\begin{align*}
\text{app}([] & , x, x) \leftarrow \\
\text{app}([x|y] & , z, [x|u]) \leftarrow \text{app}(y, z, u) \\
\text{app3}(x, y, z, u) & \leftarrow \text{app}(x, y, v), \text{app}(v, z, u)
\end{align*}
\]

\[
|\text{app}(x, y, z)| \iff \min(|x|, |z|)
\]
\[
|\text{app3}(x, y, z, u)| \iff |x| + |u| + 1
\]

shows that the program is recurrent.

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

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

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

:\iff

$I$ model of $c$,

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

$|A| > |B|$

program $P$ acceptable (w.r.t. $|\mid$ and $I$)

:\iff for some level mapping $|\mid$ and interpretation $I$, each $c \in P$ is acceptable

w.r.t. $|\mid$ and $I$
Example (I)

```prolog
app([] , x , x) ←
app([x|y] , z , [x|u]) ← app(y , z , u)
rev([], []) ←
rev([x|y] , z) ← rev(y , u) , app(u , [x] , z)
```

\[
|\text{app}(x , y , z)| \equiv \min(|x| , |z|) \\
|\text{rev}(x , y)| \equiv |x|
\]

\[I \iff \{ \text{app}(x , y , z) \mid |x| + |y| = |z| \} \]
\[\cup \{ \text{rev}(x , y) \mid |x| = |y| \}\]

shows that the program is acceptable.
Example (II)

\[
\begin{align*}
app([\ ], x, x) & \leftarrow \\
app([x|y], z, [x|u]) & \leftarrow app(y, z, u) \\
app3(x, y, z, u) & \leftarrow app(x, y, v), app(v, z, u)
\end{align*}
\]

\[
\begin{align*}
|app(x, y, z)| & :\iff |x| \\
|app3(x, y, z, u)| & :\iff |x| + |y| + 1 \\
I & :\iff \{app(x, y, z) \mid |x| + |y| = |z|\} \\
& \quad \cup ground(app3(x, y, z, u))
\end{align*}
\]

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 \( I \)
\[
\iff \\
\text{for every ground instance } \underline{A}, \underline{B}, \underline{B} \text{ of } Q \text{ such that } I \models \underline{A}, \\
|B| \leq k
\]

query \( Q \) bounded w.r.t. \( \| \) and \( I \)
\[
\iff Q \text{ bounded by some } k \text{ w.r.t. } \| \text{ and } I
\]
Example

\[
\begin{align*}
\text{app}([ ], x, x) & \leftarrow \\
\text{app}([x|y], z, [x|u]) & \leftarrow \text{app}(y, z, u) \\
\text{app}^3(x, y, z, u) & \leftarrow \text{app}(x, y, v), \text{app}(v, z, u)
\end{align*}
\]

\[
\begin{align*}
|\text{app}(x, y, z)| & \Leftrightarrow |x| \\
|\text{app}^3(x, y, z, u)| & \Leftrightarrow |x| + |y| + 1 \\
I & \Leftrightarrow \{\text{app}(x, y, z) \mid |x| + |y| = |z|\} \\
& \cup \text{ground}(\text{app}^3(x, y, z, u))
\end{align*}
\]

The program is acceptable (w.r.t. | | and I),
and \(\text{app}^3([a], [b], [c], u)\) is bounded (by \(k = 3\)) w.r.t. | | and I.
A Notational Convention

\[ \text{max: } \mathcal{P}(\mathbb{N}) \rightarrow \mathbb{N} \cup \{\omega\} \text{ with} \]

\[ \text{max } S: \Leftrightarrow \begin{cases} 0 & \text{if } S = \emptyset \\ n & \text{if } S \text{ is finite but not empty and with maximum } n \\ \omega & \text{if } S \text{ is infinite} \end{cases} \]
An Extended Notion of Boundedness (II)

Let $Q$ be a query consisting of $n \geq 1$ atoms. Then for every $i = 1, \ldots, n$ and every interpretation $I$,

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

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

$$|Q|_i : \Leftrightarrow bag(\max |Q|_1^I, \ldots, \max |Q|_n^I)$$
Example

\[
\begin{align*}
\text{app}([ \,], x, x) & \leftarrow \\
\text{app}([x|y], z, [x|u]) & \leftarrow \text{app}(y, z, u) \\
\text{app}^3(x, y, z, u) & \leftarrow \text{app}(x, y, v), \text{app}(v, z, u)
\end{align*}
\]

\[
\begin{align*}
|\text{app}(x, y, z)| & :\Leftrightarrow |x| \\
|\text{app}^3(x, y, z, u)| & :\Leftrightarrow |x| + |y| + 1 \\
l & :\Leftrightarrow \{\text{app}(x, y, z) \mid |x| + |y| = |z| \} \\
& \quad \cup \text{ground}(\text{app}^3(x, y, z, u))
\end{align*}
\]

\[
\begin{align*}
|\text{app}^3([a], [b], [c], u)|_l & = \text{bag}(3) \\
|\text{app}([a], [b], v_1), \text{app}(v_1, [c], u)|_l & = \text{bag}(1, 2)
\end{align*}
\]
Boundedness Lemma for Acceptable Programs

**Lemma 6.23**
Let $P$ be an acceptable (w.r.t. $\mid \mid$ and $I$) program. If $Q_1$ is a query bounded w.r.t. $\mid \mid$ and $I$, and if $Q_2$ is an LD-resolvent of $Q_1$, then

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

**Proof:**
1. Any instance $Q'$ of $Q$ is bounded and satisfies $|Q'|_I \leq_m |Q|_I$.
2. An instance of an acceptable clause is acceptable.
3. For every acceptable $A \leftarrow B$ and every bounded $A$, $C$, $B$, $C$ is bounded and satisfies $|B, C|_I \prec_m |A, C|_I$.

(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

\[
\begin{aligned}
app([ ], x, x) & \leftarrow \\
app([x|y], z, [x|u]) & \leftarrow app(y, z, u) \\
perm([ ], [ ]) & \leftarrow \\
perm(x, [y|z]) & \leftarrow app(u, [y|v], x), \ app(u, v, w), \ perm(w, z)
\end{aligned}
\]

\[
\begin{aligned}
|app(x, y, z)| & \equiv min(|x|, |z|) \\
|perm(x, y)| & \equiv |x| + 1 \\
I & \equiv \{app(x, y, z) \mid |x| + |y| = |z|\} \\
& \quad \cup \ ground(perm(x, y))
\end{aligned}
\]

This shows that the program is acceptable.
Objectives

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