Chapter 8

Termination of Programs
Outline

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

\begin{verbatim}
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).
\end{verbatim}
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).
ar(b, c).
ar(d, e).
connected(X, Y) :- arc(X, Y).
connected(X, Y) :- connected(X, Z), arc(Z, Y).
And this one?

```prolog
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?

```prolog
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

\begin{verbatim}
app([], X, X).
app([X|Y], Z, [X|U]) :- app(Y, Z, U).
\end{verbatim}

The query \texttt{app([a,b], Y, Z)} terminates.
The query \texttt{app(X, Y, [c,d])} terminates.
The query \texttt{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 \texttt{app3([a], [b], [c], U)} has an infinite SLD-derivation.

However, PROLOG terminates.
Multisets

multiset (written \textit{bag}(a_1, \ldots, a_n))
\[\iff\]
unordered sequence \(a_1, \ldots, a_n\)

\(\prec\) (on finite multisets of natural numbers)
\[\iff\]
\(X \prec Y\) iff \(X = (Y - \text{bag}(a)) \cup Z\)
\[\text{for some } a \in Y \text{ and } Z \text{ such that } \forall b \in Z. b < a\]

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

\textbf{Note:} \(\prec\) is irreflexive and antisymmetric
Multiset Ordering

transitive closure of a relation $R$ on a set $\mathcal{A}$
\[ \iff \]
smallest transitive relation on $\mathcal{A}$ that contains $R$

multiset ordering ($\prec_m$) $\iff$ 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 $P$ $\iff$ function $|| : HB_P \mapsto \mathbb{N}$
level of ground atom $A$ $\iff |A|$

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 $\iff$ for some level mapping $||$,
each $c \in P$ is recurrent w.r.t. $||$
Example (I)

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

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

\[ \text{subset}([x|y], z) \leftarrow \text{member}(x, z), \text{subset}(y, z) \]
\[ \text{subset}([], x) \leftarrow \]

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

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.

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

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

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

query bounded w.r.t. \( | | \) :
\( \Leftrightarrow \) all its atoms are bounded w.r.t. \( | | \)
query \( A_1, ..., A_n \) bounded by \( k \) :
\( \Leftrightarrow |A_i| \leq k \) for \( i = 1, ..., n \)

level \( |Q| \) of bounded query \( Q = A_1, ..., A_n \)
\( \Leftrightarrow \text{bag}(|A_1|, ..., |A_n|) \)
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 $|A, B, C| \preceq_m |A, H, C|$. 
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 $t$ ($|t|$)

$\iff$

$|s|t| = |t| + 1$

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

```
list([]) ←
list([x|y]) ← list(y)
```

Defining $|list(t)| \iff |t|$

shows that this program is recurrent, hence always terminating for bounded queries.
Importance of Choice of Level Mapping

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

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).
arb(b, c).
arb(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)

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

\[
\begin{align*}
|\text{app}(x, y, z)| & :\iff \min(|x|, |z|) \\
|\text{app}^3(x, y, z, u)| & :\iff |x| + |u| + 1
\end{align*}
\]

shows that the program is recurrent.

But \(\text{app}^3([a], [b], [c], u)\) is not bounded w.r.t. \(|\ |\) and indeed has an infinite derivation.

However, all LD-derivations of \(P \cup \{\text{app}^3([a], [b], [c], u)\}\) are finite.
Acceptable Programs

clause $c$ acceptable w.r.t. level mapping $| |$ 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 :
\[ \iff \text{for some level mapping $| |$ and interpretation $I$, each } c \in P \text{ is acceptable w.r.t. $| |$ and $I$} \]
Example (I)

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

\[
\begin{align*}
|app(x, \ y, \ z)| & :\Rightarrow \ min(|x|, \ |z|) \\
|rev(x, \ y)| & :\Rightarrow \ |x| \\
\mathbb{I} & :\Rightarrow \ \{app(x, \ y, \ z) \mid |x| + |y| = |z|\} \\
& \quad \cup \ \{rev(x, \ y) \mid |x| = |y|\}
\end{align*}
\]

shows that the program is acceptable.
Example (II)

\[
\text{app}([\ ], x, x) \leftarrow \\
\text{app}([x|y], z, [x|u]) \leftarrow \text{app}(y, z, u) \\
\text{app}3(x, y, z, u) \leftarrow \text{app}(x, y, v), \text{app}(v, z, u)
\]

\[
|\text{app}(x, y, z)| :\Leftrightarrow |x| \\
|\text{app}3(x, y, z, u)| :\Leftrightarrow |x| + |y| + 1
\]

\[
l :\Leftrightarrow \{\text{app}(x, y, z) \mid |x| + |y| = |z|\} \\
\cup \text{ground}(\text{app}3(x, y, z, u))
\]

shows that the program is acceptable.
Acceptability vs. Recurrence

Note 6.21

A program is recurrent w.r.t. $\mid\mid$ iff it is acceptable w.r.t. $\mid\mid$ and $HB$. 
Let $l$ be a level mapping, $I$ an interpretation, $k \in \mathbb{N}$.

query $Q$ bounded by $k$ w.r.t. $l$ and $I$:

$\iff$

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

$|B| \leq k$

query $Q$ bounded w.r.t. $l$ and $I$:

$\iff$ $Q$ bounded by some $k$ w.r.t. $l$ and $I$
Example

\[
\text{app}([], x, x) \leftarrow \\
\text{app}([x|y], z, [x|u]) \leftarrow \text{app}(y, z, u) \\
\text{app3}(x, y, z, u) \leftarrow \text{app}(x, y, v), \text{app}(v, z, u)
\]

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

The program is acceptable (w.r.t. \(||\) and \(I\)),
and \(\text{app3}([a], [b], [c], u)\) is bounded (by \(k = 3\)) w.r.t. \(||\) and \(I\).
A Notational Convention

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

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

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

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

$$|Q|_i : \iff \text{bag}(\max |Q|_1^I, \ldots, \max |Q|_n^I)$$
Example

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

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

\[
\begin{align*}
|app3([a], [b], [c], u)|_l & = bag(3) \\
|app([a], [b], v_1), app(v_1, [c], u)|_l & = bag(1, 2)
\end{align*}
\]
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 $I$
- $|Q_2|_I \preceq_m |Q_1|_I$

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

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

\[
|app(x, y, z)| \Leftrightarrow min(|x|, |z|) \\
|perm(x, y)| \Leftrightarrow |x| + 1
\]

\[
l \Leftrightarrow \{app(x, y, z) \mid |x| + |y| = |z|\} \\
\cup ground(perm(x, y))
\]

This shows that the program is acceptable.
Objectives

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