Computability properties of $\texttt{While}_{\backslash\{\texttt{cons}\}}$

Mohamed EL-AQQAD,
Guillaume Bonfante,
Mathieu Hoyrup

Institutes: Loria , Mines Nancy
mohamed.el-aqqad1@etu.univ-lorraine.fr

April 28, 2015

1. While is a simple imperative language whose data structure are abstract binary trees
2. `While` **Data** The set $\mathbb{D}$ of trees is the smallest set defined by

$$\mathbb{D} ::= \mathtt{nil} \mid (\mathbb{D} \cdot \mathbb{D})$$

Example :

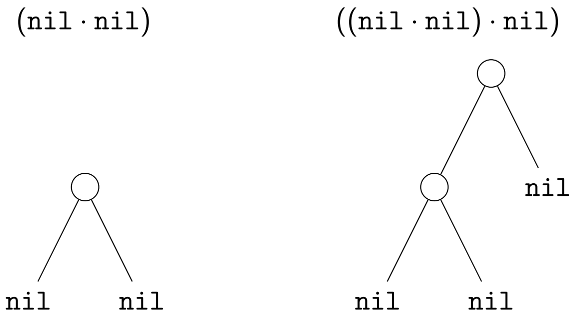


Figure 1: two elements of $\mathbb{D}$

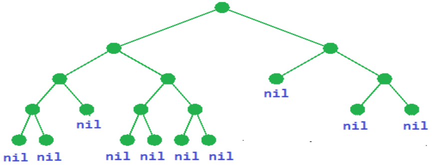
3. **Definition**

Definition (*height and length*)

We define the integer evaluated functions h the height of a tree and l its length by:

$$h(\mathtt{nil}) = 1 \forall u, v \in \mathbb{D} \quad h((u,v)) = 1 + max(h(u), h(v))$$
$$l(\mathtt{nil}) = 1 \forall u, v \in \mathbb{D} \quad l((u,v)) = l(u) + l(v)$$

4. **Example:** Consider the following tree d



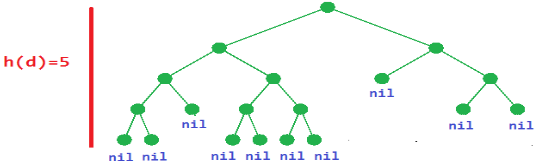
3. **Definition**

Definition (*height and length*)

We define the integer evaluated functions h the height of a tree and l its length by:

$$h(\texttt{nil}) = 1 \forall u, v \in \mathbb{D} \quad h((u, v)) = 1 + max(h(u), h(v))$$
$$l(\texttt{nil}) = 1 \forall u, v \in \mathbb{D} \quad l((u, v)) = l(u) + l(v)$$

4. **Example:** Consider the following tree d



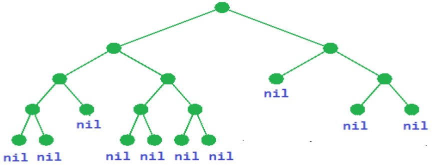
3. **Definition**

Definition (*height and length*)

We define the integer evaluated functions h the height of a tree and l its length by:

$$h(\texttt{nil}) = 1 \forall u, v \in \mathbb{D} \quad h((u, v)) = 1 + max(h(u), h(v))$$
$$l(\texttt{nil}) = 1 \forall u, v \in \mathbb{D} \quad l((u, v)) = l(u) + l(v)$$

4. **Example:** Consider the following tree d



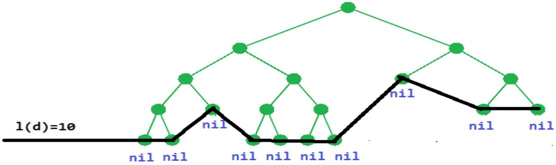
3. **Definition**

Definition (*height and length*)

We define the integer evaluated functions h the height of a tree and l its length by:

$$h(\texttt{nil}) = 1 \forall u, v \in \mathbb{D} \quad h((u,v)) = 1 + max(h(u), h(v))$$
$$l(\texttt{nil}) = 1 \forall u, v \in \mathbb{D} \quad l((u,v)) = l(u) + l(v)$$

4. **Example:** Consider the following tree d



3. **Expressions :**

Expressions	::=	X	#A variable
	|	t	#A constant
	|	hd E	#The first projection
			#e.g hd((u,v))==u
	|	tl E	#The second projection
			#e.g tl((u,v))==v
	|	cons E F	#The tree constructors
			#e.g cons u v == (u,v)
	|	E =? F	#Test of the equality
			#e.g (u=?v)==nil iff u≠v

4. **Commands :**

$$
\begin{array}{llll}
\texttt{Commands} & ::= & \texttt{X} := \texttt{E} & \texttt{\# Assignement} \\
 & | & \texttt{C ; D} & \texttt{\# Sequence of two commands} \\
 & | & \texttt{while (E) do \{ C \}} & \texttt{\# The while loop}
\end{array}
$$

5. **Programs :** the set of programs $\mathbb{P}$ contains all elements of the form:

$$\texttt{read X}_1, \ldots, \texttt{X}_n;\ \texttt{C};\ \texttt{write Y}$$

where $\texttt{C}$ is a command and $\texttt{X}_1, \ldots, \texttt{X}_n, \texttt{Y}$ are variables.

Example :

$$\mathbf{p} \triangleq \texttt{read X}_1, \texttt{X}_2;\ \texttt{Y} := \texttt{X}_1;\ \texttt{write Y}$$

we write then $\forall t_1, t_2 \in \mathbb{D} \ \ [\![p]\!](t_1, t_2) = t_1$.

The Language $\texttt{While}_{\backslash\{\texttt{cons}\}}$

- Origin of $\texttt{While}_{\backslash\{\texttt{cons}\}}$
  - Considered by Neil Jones to establish some complexity properties in `while`
  - We continue the study of this Language because of its interesting properties.
- $\texttt{While}_{\backslash\{\texttt{cons}\}}$ is a restricted language of `While` which does not contain the tree constructors, the subset of its expressions is:

Expressions	::=	X	#A variable
	|	t	#A constant
	|	hd E	#the first projection
	|	tl E	#the second projection
	|	E =? F	#The test of equality

Implicit complexity

1. **Earlier by Neil Jones :** As we already noticed the first results in complexity about $\texttt{While}_{\backslash\{\text{cons}\}}$ was proved by Neil Jones.
   - $\texttt{While}_{\backslash\{\text{cons}\}}$

Theorem (*Neil Jones 1997*)

$\texttt{While}_{\backslash\{\text{cons}\}}$ *is LOGSPACE-complete, i.e it computes exactly functions which can be done in a logarithmic amount of space on $I(d)$.*

   - $\texttt{While}_{\backslash\{\text{cons}\}}$ **with recursion**

Theorem (*Neil Jones 1997*)

The set of computable functions in $\texttt{While}_{\backslash\{\text{cons}\}}$ *with recursion is identical to the set of functions computable in PTIME on $I(d)$.*

In computer virology

1. $\texttt{While}_{\backslash\{\texttt{cons}\}}$ **is not acceptable:**

Theorem (*CIE 2015*)

$\texttt{While}_{\backslash\{\texttt{cons}\}}$ *has no* smh *program.*

2. **Klenne's theorem in** $\texttt{While}_{\backslash\{\texttt{cons}\}}$ **:**

Theorem (*CIE 2015*)

The Kleene's fix-point theorem does not hold in $\texttt{While}_{\backslash\{\texttt{cons}\}}$.

3. **Consequence** It is hard to write viruses in this language.

Today: the equality problem

1. **Motivation**
   - Let's consider the set of expressions in `While` :

$$\texttt{X} \;\; | \; \texttt{t} \;\; | \;\; \texttt{hd E} \;\; | \;\; \texttt{tl E} \;\; | \;\; \texttt{cons E F} \;\; \texttt{E =? F}$$

Today: the equality problem

1. **Motivation**
   - Let's consider the set of expressions in `While` :

`X | t | hd E | tl E | cons E F | E =? F`

Atomic expressions $\Longrightarrow$ constant time computing

Today: the equality problem

1. **Motivation**
   - Let's consider the set of expressions in `While` :

X	t	hd E	tl E	cons E F	E =? F

Atomic expressions $\Longrightarrow$ constant time computing

Not atomic expression

Today: the equality problem

1. **Motivation**
   - Let's consider the set of expressions in `While` :

     `X | t | hd E | tl E | cons E F` `E =? F`

     Atomic expressions $\Longrightarrow$ constant time computing
     Not atomic expression
   - Why not the **atomic equality E? = nil** which can be clearly computed in a constant time?

Today: the equality problem

1. **Motivation**
   - Let's consider the set of expressions in `While` :

X	t	hd E	tl E	cons E F	E =? F

Atomic expressions $\Longrightarrow$ constant time computing
Not atomic expression

   - Why not the **atomic equality E? = nil** which can be clearly computed in a constant time?

2. **In `while`**

Theorem (*Jones 1999*)

We can compute the equality in `While`*, this means that there exists a program* **p** *in* `While` *using only atomic expressions such that:*

$$[\![\mathbf{p}]\!](\mathtt{t},\mathtt{t}') = \mathit{nil} \qquad \textit{if } \mathtt{t} \neq \mathtt{t}' \textit{ and } (\mathit{nil},\mathit{nil}) \textit{ otherwise}$$

The equality problem

3. Using tree constructors $\Longrightarrow$ Computing equality in $\texttt{While}$
4. But what happens in $\texttt{While}_{\backslash\{\texttt{cons}\}}$?
5. **The language $\texttt{While\_Atomic}_{\backslash\{\texttt{cons}\}} \subseteq \texttt{While}_{\backslash\{\texttt{cons}\}}$**
   We consider the language $\texttt{While\_Atomic}_{\backslash\{\texttt{cons}\}}$ which uses only atomic expressions, in other terms:

$$\texttt{Expressions} ::= \texttt{X} \mid \texttt{t} \mid \texttt{hd E} \mid \texttt{tl E} \mid \texttt{E =?nil}$$

6. Do they compute the same functions $\texttt{While\_Atomic}_{\backslash\{\texttt{cons}\}} \simeq \texttt{While}_{\backslash\{\texttt{cons}\}}$?

The equality problem

7. Let us try to compute equality in $\texttt{While\_Atomic}_{\backslash\{\texttt{cons}\}}$.
8. **Formulation of the problem**

Question.

Is there a program **p** in $\texttt{While\_Atomic}_{\backslash\{\texttt{cons}\}}$ such that:

$$[\![\mathbf{p}]\!](\mathtt{t},\mathtt{t}') = \mathtt{nil} \qquad \text{if } \mathtt{t} \neq \mathtt{t}' \text{ and } (\mathtt{nil},\mathtt{nil}) \text{ otherwise ?}$$

The equality problem

- **The equality in $\texttt{While}_{\backslash\{\texttt{cons}\}}$**

Theorem

There is no program which computes the equality in $\texttt{While\_Atomic}_{\backslash\{\texttt{cons}\}}$

- **sketch of the proof**
- For simplification, we will use the following equivalences :

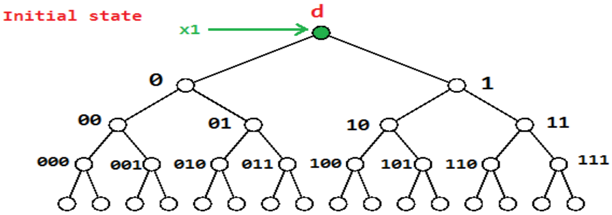
$$\begin{aligned} \texttt{X}_i := \texttt{hd}\texttt{X}_j &\equiv \quad \texttt{X}_i := \overline{\texttt{0}}\texttt{X}_j \\ \texttt{X}_i := \texttt{tl}\texttt{X}_j &\equiv \quad \texttt{X}_i := \overline{\texttt{1}}\texttt{X}_j \\ \Longrightarrow \ x_i := \texttt{hd tl tl tl X}_j &\equiv \ x_i := \overline{\texttt{0111}}\texttt{X}_j \end{aligned}$$

Definition

Given an element $d \in \mathbb{D}$ and a program p we say that a word $w \in \Sigma = \{0, 1\}^*$ is visited if there exists a variable x_i such that $value(x_i) = \overline{w}(d)$ at some moment of the execution of the program p over the input d .

The equality problem

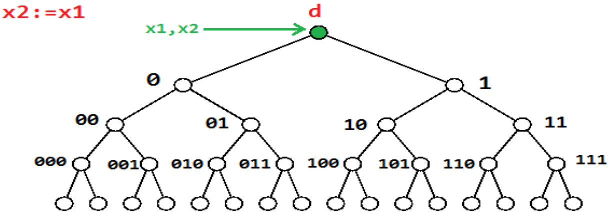
- **Example:**



$$
\begin{aligned}
\mathbf{p} \triangleq\ & \mathtt{read}\ \mathtt{X}_1;\\
& \mathtt{X}_2 := \mathtt{X}_1;\\
& \mathtt{X}_2 := \mathtt{hd}\ \ \mathtt{X}_1;\\
& \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1;\\
& \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_2;\\
& \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1;\\
& \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1;\\
& \mathtt{write}\ \mathtt{X}_1
\end{aligned}
$$

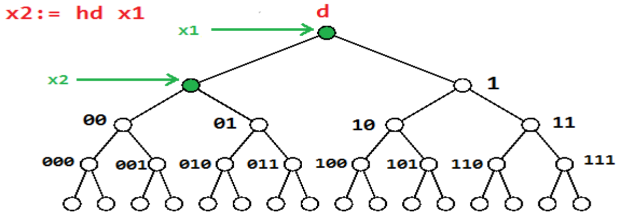
The equality problem

- **Example:**



$\mathbf{p} \triangleq$ `read` $\mathtt{X}_1$;
$\mathtt{X}_2 := \mathtt{X}_1$;
$\mathtt{X}_2 := \mathtt{hd}\ \ \mathtt{X}_1$;
$\mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1$;
$\mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_2$;
$\mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1$;
$\mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1$;
`write` $\mathtt{X}_1$

- **Example:**



$\mathbf{p} \triangleq \texttt{read}\ X_1;$

$X_2 := X_1;$

$X_2 := \texttt{hd}\ X_1;$

$X_1 := \texttt{tl}\ X_1;$

$X_1 := \texttt{tl}\ X_2;$

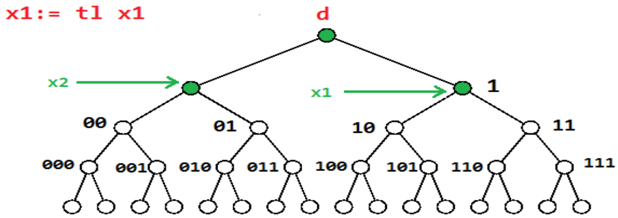
$X_1 := \texttt{tl}\ X_1;$

$X_1 := \texttt{tl}\ X_1;$

$\texttt{write}\ X_1$

The equality problem

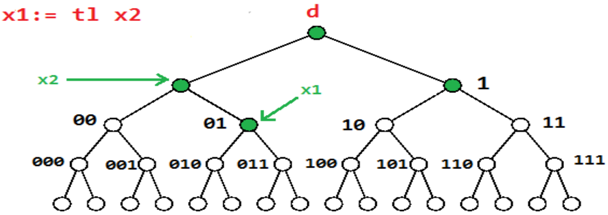
- **Example:**



$\mathbf{p} \triangleq$ `read` X_1;
$X_2 := X_1$;
$X_2 :=$ `hd` X_1;
$X_1 :=$ `tl` X_1;
$X_1 :=$ `tl` X_2;
$X_1 :=$ `tl` X_1;
$X_1 :=$ `tl` X_1;
`write` X_1

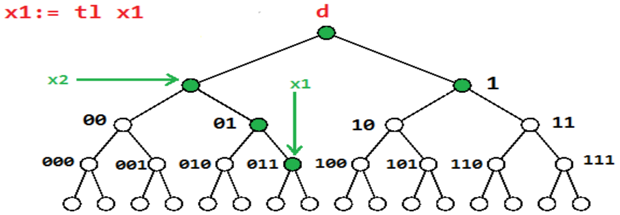
The equality problem

- **Example:**



$\mathbf{p} \triangleq$ `read` X_1;
$X_2 := X_1$;
$X_2 :=$ `hd` X_1;
$X_1 :=$ `tl` X_1;
$X_1 :=$ `tl` X_2;
$X_1 :=$ `tl` X_1;
$X_1 :=$ `tl` X_1;
`write` X_1

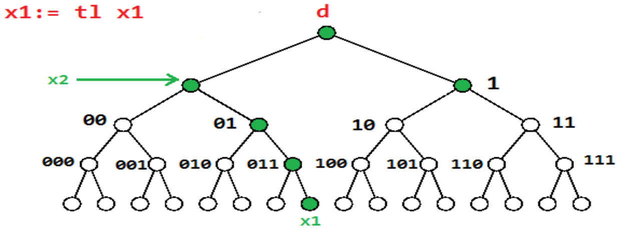
- **Example:**



$$\begin{aligned} \mathbf{p} \triangleq\ & \mathtt{read}\ \mathtt{X}_1; \\ & \mathtt{X}_2 := \mathtt{X}_1; \\ & \mathtt{X}_2 := \mathtt{hd}\ \ \mathtt{X}_1; \\ & \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1; \\ & \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_2; \\ & \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1; \\ & \mathtt{X}_1 := \mathtt{tl}\ \ \mathtt{X}_1; \\ & \mathtt{write}\ \mathtt{X}_1 \end{aligned}$$

The equality problem

- **Example:**



$\mathbf{p} \triangleq \texttt{read}\ \texttt{X}_1;$

$\texttt{X}_2 := \texttt{X}_1;$

$\texttt{X}_2 := \texttt{hd}\ \ \texttt{X}_1;$

$\texttt{X}_1 := \texttt{tl}\ \ \texttt{X}_1;$

$\texttt{X}_1 := \texttt{tl}\ \ \texttt{X}_2;$

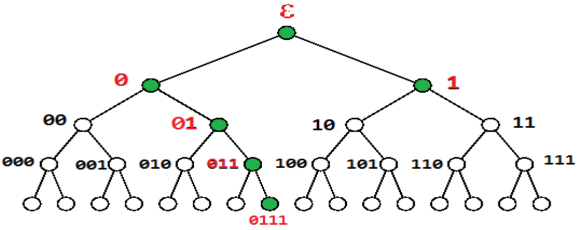
$\texttt{X}_1 := \texttt{tl}\ \ \texttt{X}_1;$

$\texttt{X}_1 := \texttt{tl}\ \ \texttt{X}_1;$

$\texttt{write}\ \texttt{X}_1$

The equality problem

- **Example:**



$$\mathbf{p} \triangleq \texttt{read } \texttt{X}_1;$$
$$\texttt{X}_2 := \texttt{X}_1;$$
$$\texttt{X}_2 := \texttt{hd } \texttt{X}_1;$$
$$\texttt{X}_1 := \texttt{tl } \texttt{X}_1;$$
$$\texttt{X}_1 := \texttt{tl } \texttt{X}_2;$$
$$\texttt{X}_1 := \texttt{tl } \texttt{X}_1;$$
$$\texttt{X}_1 := \texttt{tl } \texttt{X}_1;$$
$$\texttt{write } \texttt{X}_1$$

- The set of visited points is

$$S(\mathbf{p}, d) = \{\epsilon, 0, 1, 01, 011, 0111\}$$

The equality problem

Proposition

For any given program $\mathbf{p}$ *there exists a polynomial* P *such that for any complete binary tree* d *of height* h *the number of visited points* $|S(\mathbf{p}, d)|$ *is less than or equal to* $P(h)$.

- The graph of all possible states of an executing program has vertices the value of tuple $(\mathtt{X}_1, \cdots, \mathtt{X}_m, L)$ where $\mathtt{X}_1, \cdots, \mathtt{X}_m$ are variables and L a location in the program.
- The number of possible values of the tuple $(\mathtt{X}_1, \cdots, \mathtt{X}_m, L)$ is $l(h+1)^m$, m is the number of variables and l the length of the program.
- A walk in such a graph is either a path or a cycle and the number of visited points is less than

$$|S(\mathbf{p}, d)| \leq P(h) = l(h+1)^m$$

proof of the principal theorem

changing slightly the inputs $\Longrightarrow$ the same output

Assume that the equality is computable in $\texttt{While\_Atomic}_{\setminus\{\texttt{cons}\}}$ so that there exists a program **p** which compute the equality, and let d be a perfect binary tree of height $h = h(d)$.

- First we have $[\![\mathbf{p}]\!](d, d) = (\texttt{nil}, \texttt{nil})$
- The number of visited points $|S(p, d)|$ is polynomial on h
- The number of leafs of d is 2^h
- There exists a non visited leaf in d
- It turns out that if we insert an element in a non-visited leaf of d we will have the same flow of execution
- Let d' be the element obtain from d by changing the non-visited leaf of d by (nil, nil) we will have

$$[\![\mathbf{p}]\!](d', d) = (\texttt{nil}, \texttt{nil}) \quad \text{and } d' \neq d$$

Thank you for your attention