Script generated by TTT

Title: Seidl: Programmoptimierung (21.11.2012)

Date: Wed Nov 21 09:34:01 CET 2012

Duration: 87:16 min

Pages: 39

$M : (\mathbb{N} \to \mathbb{Z}^{\top})_{fin}$

(3) Constant Propagation:

- Extend the abstract state by an abstract store M
- Execute accesses to known memory locations!

$$[\![x = M[e];]\!]^\sharp\,(D, M) = \begin{cases} (D \oplus \{x \mapsto M\,a\}, M) & \text{if } [\![e]\!]^\sharp\,D = a \sqsubset \top \\ (D \oplus \{x \mapsto \top\}, M) & \text{otherwise} \end{cases}$$

$$[\![M[e_1] = e_2;]\!]^\sharp\,(D, M) = \begin{cases} (D, M \oplus \{a \mapsto [\![e_2]\!]^\sharp D\}) & \text{if } [\![e_1]\!]^\sharp\,D = a \sqsubset \top \\ (D, \underline{\top}) & \text{otherwise} \quad \text{where} \end{cases}$$

$$\underline{\top}\,a = \top \qquad (a \in \mathbb{N})$$

$M = \{a_1 \mapsto d_1, \ldots, a_2 \mapsto d_2\}$

Problems:

- Addresses are from $\mathbb{N}$:-(
  There are no infinite strictly ascending chains, but ...
- Exact addresses at compile-time are rarely known :-(
- At the same program point, typically different addresses are accessed ...
- Storing at an unknown address destroys all information M :-(

$\Longrightarrow$ constant propagation fails :-(

$\Longrightarrow$ memory accesses/pointers kill precision :-(

Simplification:

- We consider pointers to the beginning of blocks A which allow indexed accesses $A[i]$:-)
- We ignore well-typedness of the blocks.
- New statements:

$x = \mathsf{new}();$ // allocation of a new block

$x = y[e];$ // indexed read access to a block

$y[e_1] = e_2;$ // indexed write access to a block

- Blocks are possibly infinite :-)
- For simplicity, all pointers point to the beginning of a block.

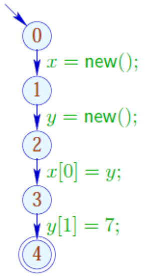
Simple Example:

$x = \mathsf{new}();$

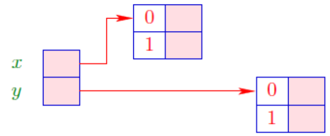
$y = \mathsf{new}();$

$x[0] = y;$

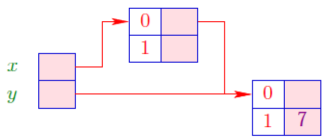
$y[1] = 7;$



The Semantics:



The Semantics:



More Complex Example:

$r = \mathsf{Null};$

while $(t \neq \mathsf{Null})$ {

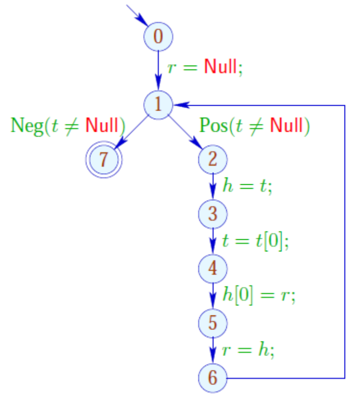
$\quad h = t;$

$\quad t = t[0];$

$\quad h[0] = r;$

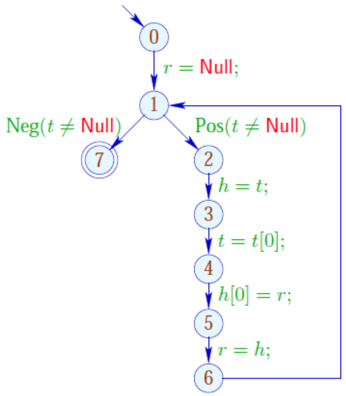
$\quad r = h;$

}



More Complex Example:

$r = \text{Null};$
while $(t \neq \text{Null})$ {
    $h = t;$
    $t = t[0];$
    $h[0] = r;$
    $r = h;$
}



Concrete Semantics:

A store consists of a finite collection of blocks.

After h new-operations we obtain:

$$
\begin{array}{lcll}
Addr_h & = & \{\text{ref } a \mid 0 \leq a < h\} & // \text{ addresses} \\
Val_h & = & Addr_h \cup \mathbb{Z} & // \text{ values} \\
Store_h & = & (Addr_h \times \mathbb{N}_0) \to Val_h & // \text{ store} \\
State_h & = & (Vars \to Val_h) \times Store_h & // \text{ states}
\end{array}
$$

For simplicity, we set: $0 = \text{Null}$

Concrete Semantics:

A store consists of a finite collection of blocks.

After h new-operations we obtain:

$$
\begin{array}{lcll}
Addr_h & = & \{\text{ref } a \mid 0 \leq a < h\} & // \text{ addresses} \\
Val_h & = & Addr_h \cup \mathbb{Z} & // \text{ values} \\
Store_h & = & \cancel{(Addr_h} \times \mathbb{N}_0) \to Val_h & // \text{ store} \\
State_h & = & (Vars \to Val_h) \times Store_h & // \text{ states}
\end{array}
$$

For simplicity, we set: $0 = \text{Null}$

373

Let $(\rho, \mu) \in State_h$. Then we obtain for the new edges:

h

$$
\begin{array}{lcl}
[\![x = \text{new}();]\!]\,(\rho, \mu) & = & (\rho \oplus \{x \mapsto \text{ref } h\}, \\
 & & \mu \oplus \{(\text{ref } h, i) \mapsto 0 \mid i \in \mathbb{N}_0\}) \\
[\![x = y[e];]\!]\,(\rho, \mu) & = & (\rho \oplus \{x \mapsto \mu\,(\rho\,y, [\![e]\!]\,\rho)\}, \mu) \\
[\![y[e_1] = e_2;]\!]\,(\rho, \mu) & = & (\rho, \mu \oplus \{(\rho\,y, [\![e_1]\!]\,\rho) \mapsto [\![e_2]\!]\,\rho\})
\end{array}
$$

h+1

Concrete Semantics:

A store consists of a finite collection of blocks.

After h new-operations we obtain:

$$\begin{array}{lcll} Addr_h & = & \{\mathsf{ref}\ a \mid 0 \leq a < h\} & // \text{ addresses} \\ Val_h & = & Addr_h \cup \mathbb{Z} & // \text{ values} \\ Store_h & = & (Addr_h \times \mathbb{N}_0) \to Val_h & // \text{ store} \\ State_h & = & (Vars \to Val_h) \times Store_h & // \text{ states} \end{array}$$

For simplicity, we set: $0 = \mathsf{Null}$

Let $(\rho, \mu) \in State_h$. Then we obtain for the new edges:

$$\begin{array}{lcl} \llbracket x = \mathsf{new}(); \rrbracket\ (\rho, \mu) & = & (\rho \oplus \{x \mapsto \mathsf{ref}\ h\}, \\ & & \quad \mu \oplus \{(\mathsf{ref}\ h, i) \mapsto 0 \mid i \in \mathbb{N}_0\}) \\ \llbracket x = y[e]; \rrbracket\ (\rho, \mu) & = & (\rho \oplus \{x \mapsto \mu\,(\rho\, y, \llbracket e \rrbracket\, \rho)\}, \mu) \\ \llbracket y[e_1] = e_2; \rrbracket\ (\rho, \mu) & = & (\rho, \mu \oplus \{(\rho\, y, \llbracket e_1 \rrbracket\, \rho) \mapsto \llbracket e_2 \rrbracket\, \rho\}) \end{array}$$

Concrete Semantics:

A store consists of a finite collection of blocks.

After h new-operations we obtain:

$$\begin{array}{lcll} Addr_h & = & \{\mathsf{ref}\ a \mid 0 \leq a < h\} & // \text{ addresses} \\ Val_h & = & Addr_h \cup \mathbb{Z} & // \text{ values} \\ Store_h & = & (Addr_h \times \mathbb{N}_0) \to Val_h & // \text{ store} \\ State_h & = & (Vars \to Val_h) \times Store_h & // \text{ states} \end{array}$$

For simplicity, we set: $0 = \mathsf{Null}$

373

Caveat:

This semantics is too detailled in that it computes with absolute Addresses. Accordingly, the two programs:

$$\begin{array}{lcl} x = \mathsf{new}(); & \qquad & y = \mathsf{new}(); \\ y = \mathsf{new}(); & \qquad & x = \mathsf{new}(); \end{array}$$

are not considered as equivalent !!?

Possible Solution:

Define equivalence only up to permutation of addresses :-)

Alias Analysis 1. Idea:

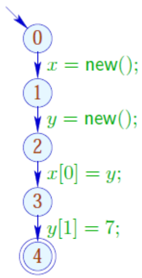
- Distinguish finitely many classes of blocks.
- Collect all addresses of a block into one set!
- Use sets of addresses as abstract values!

$\Longrightarrow$ Points-to-Analysis

$$
\begin{array}{llll}
\mathit{Addr}^\sharp & = & \mathit{Edges} & // \text{ creation edges} \\
\mathit{Val}^\sharp & = & 2^{\mathit{Addr}^\sharp} & // \text{ abstract values} \\
\mathit{Store}^\sharp & = & \mathit{Addr}^\sharp \to \mathit{Val}^\sharp & // \text{ abstract store} \\
\mathit{State}^\sharp & = & (\mathit{Vars} \to \mathit{Val}^\sharp) \times \mathit{Store}^\sharp & // \text{ abstract states}
\end{array}
$$

// complete lattice !!!

... in the Simple Example:



	x	y	$(0,1)$
0	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{(0,1)\}$	$\emptyset$	$\emptyset$
2	$\{(0,1)\}$	$\{(1,2)\}$	$\emptyset$
3	$\{(0,1)\}$	$\{(1,2)\}$	$\{(1,2)\}$
4	$\{(0,1)\}$	$\{(1,2)\}$	$\{(1,2)\}$

The Effects of Edges:

$$
\begin{array}{lll}
[\![(\_, ;, \_)]\!]^\sharp\,(D, M) & = & (D, M) \\
[\![(\_, \mathrm{Pos}(e), \_)]\!]^\sharp\,(D, M) & = & (D, M) \\
[\![(\_, x = y;, \_)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto D\,y\}, M) \\
[\![(\_, x = e;, \_)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto \emptyset\}, M) \qquad , \qquad e \notin \mathit{Vars} \\
\\
[\![(u, x = \mathsf{new}();, v)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto \{(u,v)\}\}, M) \\
[\![(\_, x = y[e];, \_)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto \bigcup\{M(f) \mid f \in D\,y\}\}, M) \\
[\![(\_, y[e_1] = x;, \_)]\!]^\sharp\,(D, M) & = & (D, M \oplus \{f \mapsto (M\,f \cup D\,x) \mid f \in D\,y\})
\end{array}
$$

ref(e, s) Δ e

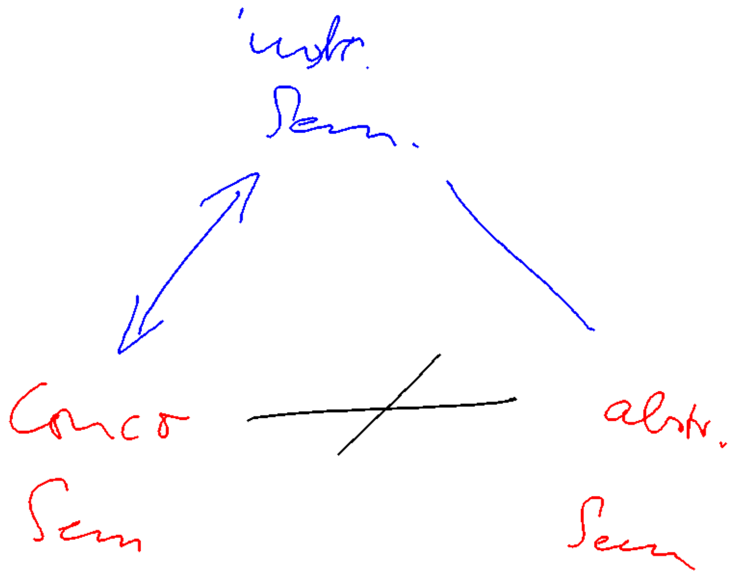
Caveat:

- The value Null has been ignored. Dereferencing of Null or negative indices are not detected :-(
- Destructive updates are only possible for variables, not for blocks in storage!

  $\Longrightarrow$ no information, if not all block entries are initialized before use :-((
- The effects now depend on the edge itself.

  The analysis cannot be proven correct w.r.t. the reference semantics :-(

  In order to prove correctness, we first instrument the concrete semantics with extra information which records where a block has been created.



...

- We compute possible points-to information.
- From that, we can extract may-alias information.
- The analysis can be rather expensive — without finding very much :-(
- Separate information for each program point can perhaps be abandoned ??

The Effects of Edges:

$$
\begin{array}{lcll}
[\![(\_, ;, \_)]\!]^\sharp\,(D, M) & = & (D, M) & \\
[\![(\_, \mathrm{Pos}(e), \_)]\!]^\sharp\,(D, M) & = & (D, M) & \\
[\![(\_, x = y;, \_)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto D\,y\}, M) & \\
[\![(\_, x = e;, \_)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto \emptyset\}, M) & , \quad e \notin \mathit{Vars} \\
 & & & \\
[\![(u, x = \mathsf{new}();, v)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto \{(u, v)\}\}, M) & \\
[\![(\_, x = y[e];, \_)]\!]^\sharp\,(D, M) & = & (D \oplus \{x \mapsto \bigcup\{M(f) \mid f \in D\,y\}\}, M) & \\
[\![(\_, y[e_1] = x;, \_)]\!]^\sharp\,(D, M) & = & (D, M \oplus \{f \mapsto (M\,f \cup D\,x) \mid f \in D\,y\}) &
\end{array}
$$

...

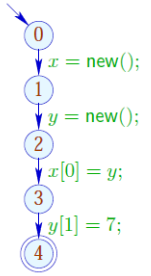
- We compute possible points-to information.
- From that, we can extract may-alias information.
- The analysis can be rather expensive — without finding very much :-(
- Separate information for each program point can perhaps be abandoned ??

380

Alias Analysis 2. Idea:

Compute for each variable and address a value which safely approximates the values at every program point simultaneously !

... in the Simple Example:



x	$\{(0,1)\}$
y	$\{(1,2)\}$
$(0,1)$	$\{(1,2)\}$
$(1,2)$	$\emptyset$

Each edge (u, lab, v) gives rise to constraints:

lab	*Constraint*
$x = y;$	$\mathcal{P}[x] \supseteq \mathcal{P}[y]$
$x = \mathsf{new}();$	$\mathcal{P}[x] \supseteq \{(u,v)\}$
$x = y[e];$	$\mathcal{P}[x] \supseteq \bigcup\{\mathcal{P}[f] \mid f \in \mathcal{P}[y]\}$
$y[e_1] = x;$	$\mathcal{P}[f] \supseteq (f \in \mathcal{P}[y]) \,?\, \mathcal{P}[x] : \emptyset$ for all $f \in Addr^\sharp$

Other edges have no effect :-)

Caveat:

This semantics is too detailled in that it computes with absolute Addresses. Accordingly, the two programs:

$$x = \mathsf{new}(); \qquad y = \mathsf{new}();$$
$$y = \mathsf{new}(); \qquad x = \mathsf{new}();$$

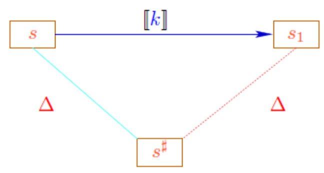
are not considered as equivalent !!?

Possible Solution:

Define equivalence only up to permutation of addresses :-)

Discussion:

- The resulting constraint system has size $\mathcal{O}(k \cdot n)$ for k abstract addresses and n edges :-(
- The number of necessary iterations is $\mathcal{O}(k + \#Vars)$ ...
- The computed information is perhaps still too zu precise !!?
- In order to prove correctness of a solution $s^\sharp \in States^\sharp$ we show:



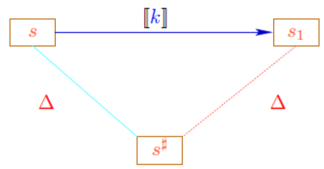
Each edge (u, lab, v) gives rise to constraints:

lab	$Constraint$
$x = y;$	$\mathcal{P}[x] \supseteq \mathcal{P}[y]$
$x = \mathsf{new}();$	$\mathcal{P}[x] \supseteq \{(u, v)\}$
$x = y[e];$	$\mathcal{P}[x] \supseteq \bigcup\{\mathcal{P}[f] \mid f \in \mathcal{P}[y]\}$
$y[e_1] = x;$	$\mathcal{P}[f] \supseteq (f \in \mathcal{P}[y])\, ?\, \mathcal{P}[x] : \emptyset$ for all $f \in Addr^\sharp$

Other edges have no effect :-)

Discussion:

- The resulting constraint system has size $\mathcal{O}(k \cdot n)$ for k abstract addresses and n edges :-(
- The number of necessary iterations is $\mathcal{O}(k + \# Vars)$ ...
- The computed information is perhaps still too zu precise !!?
- In order to prove correctness of a solution $s^\sharp \in States^\sharp$ we show:



Each edge (u, lab, v) gives rise to constraints:

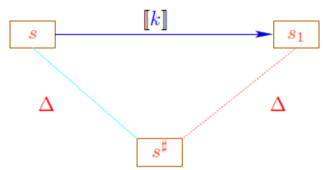
lab	$Constraint$
$x = y;$	$\mathcal{P}[x] \supseteq \mathcal{P}[y]$
$x = \mathsf{new}();$	$\mathcal{P}[x] \supseteq \{(u, v)\}$
$x = y[e];$	$\mathcal{P}[x] \supseteq \bigcup\{\mathcal{P}[f] \mid f \in \mathcal{P}[y]\}$
$y[e_1] = x;$	$\mathcal{P}[f] \supseteq (f \in \mathcal{P}[y])\, ?\, \mathcal{P}[x] : \emptyset$ for all $f \in Addr^\sharp$

Other edges have no effect :-)

$k(k + \# Vars)$

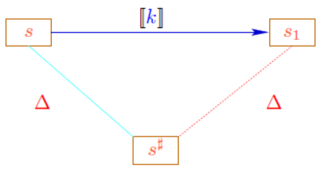
Discussion:

- The resulting constraint system has size $\mathcal{O}(k \cdot n)$ for k abstract addresses and n edges :-(
- The number of necessary iterations is $\mathcal{O}(k + \# Vars)\, k$ ...
- The computed information is perhaps still too zu precise !!?
- In order to prove correctness of a solution $s^\sharp \in States^\sharp$ we show:



Discussion:

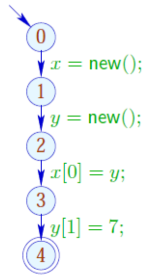
- The resulting constraint system has size $\mathcal{O}(k \cdot n)$ for k abstract addresses and n edges :-(
- The number of necessary iterations is $\mathcal{O}(k + \#Vars)$ ...
- The computed information is perhaps still too zu precise !!?
- In order to prove correctness of a solution $s^\sharp \in States^\sharp$ we show:



Alias Analysis 3. Idea:

Determine one equivalence relation $\equiv$ on variables x and memory accesses $y[\,]$ with $s_1 \equiv s_2$ whenever s_1, s_2 may contain the same address at some u_1, u_2

... in the Simple Example:



$$\equiv \;=\; \{\{x\},\ \{y, x[\,]\},\ \{y[\,]\}\}$$

Discussion:

→ We compute a single information fo the whole program.

→ The computation of this information maintains partitions $\pi = \{P_1, \ldots, P_m\}$:-)

→ Individual sets P_i are identified by means of representatives $p_i \in P_i$.

→ The operations on a partition π are:

$$\begin{array}{lll} \text{find}\,(\pi, p) & = & p_i \qquad \text{if } p \in P_i \\ & // & \text{returns the representative} \\ \text{union}\,(\pi, p_{i_1}, p_{i_2}) & = & \{P_{i_1} \cup P_{i_2}\} \cup \{P_j \mid i_1 \neq j \neq i_2\} \\ & // & \text{unions the represented classes} \end{array}$$

Discussion:

→ We compute a single information fo the whole program.

→ The computation of this information maintains partitions $\pi = \{P_1, \ldots, P_m\}$:-)

→ Individual sets P_i are identified by means of representatives $p_i \in P_i$.

→ The operations on a partition π are:

$$\begin{array}{lll} \text{find}\,(\pi, p) & = & p_i \qquad \text{if } p \in P_i \\ & // & \text{returns the representative} \\ \text{union}\,(\pi, p_{i_1}, p_{i_2}) & = & \{P_{i_1} \cup P_{i_2}\} \cup \{P_j \mid i_1 \neq j \neq i_2\} \\ & // & \text{unions the represented classes} \end{array}$$

385

→ If $x_1, x_2 \in \mathit{Vars}$ are equivalent, then also $x_1[\,]$ and $x_2[\,]$ must be equivalent :-)

→ If $P_i \cap \mathit{Vars} \neq \emptyset$, then we choose $p_i \in \mathit{Vars}$. Then we can apply union recursively :

$$\begin{array}{lll} \mathsf{union}^*\,(\pi, q_1, q_2) & = & \mathsf{let}\;\; p_{i_1} = \mathsf{find}\,(\pi, q_1) \\ & & \qquad\;\; p_{i_2} = \mathsf{find}\,(\pi, q_2) \\ & & \mathsf{in}\;\; \mathsf{if}\; p_{i_1} == p_{i_2}\; \mathsf{then}\; \pi \\ & & \qquad \mathsf{else}\;\; \mathsf{let}\; \pi = \mathsf{union}\,(\pi, p_{i_1}, p_{i_2}) \\ & & \qquad\qquad \mathsf{in}\;\; \mathsf{if}\; p_{i_1}, p_{i_2} \in \mathit{Vars}\; \mathsf{then} \\ & & \qquad\qquad\qquad \mathsf{union}^*\,(\pi, p_{i_1}[\,], p_{i_2}[\,]) \end{array}$$

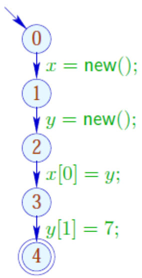
The analysis iterates over all edges once:

$$\begin{array}{l} \pi = \{\{x\}, \{x[\,]\} \mid x \in \mathit{Vars}\}; \\ \mathsf{forall}\;\; k = (\_, lab, \_)\;\; \mathsf{do}\;\; \pi = [\![lab]\!]^\sharp\,\pi; \end{array}$$

where:

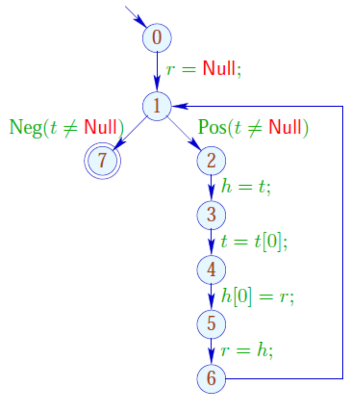
$$\begin{array}{lll} [\![x = y;]\!]^\sharp\,\pi & = & \mathsf{union}^*\,(\pi, x, y) \\ [\![x = y[e];]\!]^\sharp\,\pi & = & \mathsf{union}^*\,(\pi, x, y[\,]) \\ [\![y[e] = x;]\!]^\sharp\,\pi & = & \mathsf{union}^*\,(\pi, x, y[\,]) \\ [\![lab]\!]^\sharp\,\pi & = & \pi \qquad\qquad \text{otherwise} \end{array}$$

... in the Simple Example:



	$\{\{x\},\{y\},\{x[\,]\},\{y[\,]\}\}$
$(0,1)$	$\{\{x\},\{y\},\{x[\,]\},\{y[\,]\}\}$
$(1,2)$	$\{\{x\},\{y\},\{x[\,]\},\{y[\,]\}\}$
$(2,3)$	$\{\{x\},\boxed{\{y,x[\,]\}},\{y[\,]\}\}$
$(3,4)$	$\{\{x\},\{y,x[\,]\},\{y[\,]\}\}$

... in the More Complex Example:



	$\{\{h\},\{r\},\{t\},\{h[\,]\},\{t[\,]\}\}$
$(2,3)$	$\{\boxed{\{h,t\}},\{r\},\boxed{\{h[\,],t[\,]\}}\}$
$(3,4)$	$\{\boxed{\{h,t,h[\,],t[\,]\}},\{r\}\}$
$(4,5)$	$\{\boxed{\{h,t,r,h[\,],t[\,]\}}\}$
$(5,6)$	$\{\{h,t,r,h[\,],t[\,]\}\}$