Shape Analysis

Goal: to obtain a finite representation of the shape of the heap of a language with pointers.

The analysis result can be used for

- detection of pointer aliasing
- detection of sharing between structures
- software development tools
  – detection of errors like dereferences of `nil`-pointers
- program verification
  – `reverse` transforms a non-cyclic list to a non-cyclic list

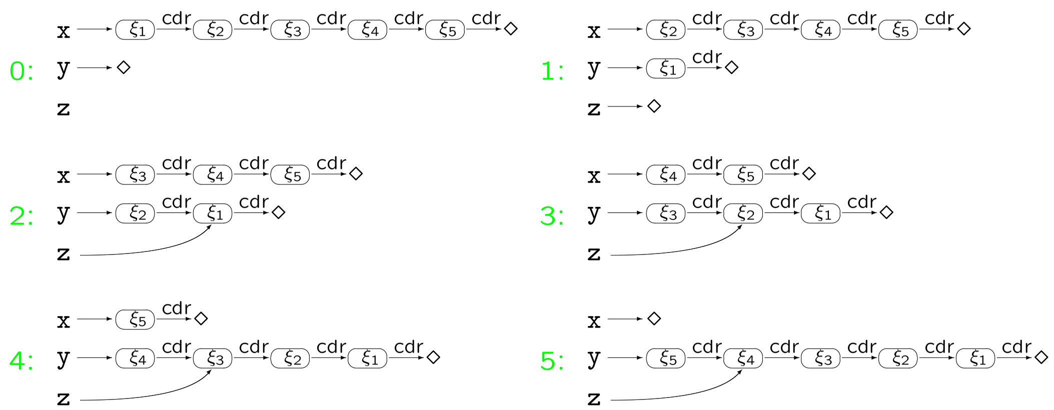
Syntax of the pointer language

$$\begin{array}{lcl} a & ::= & p \mid n \mid a_1\ op_a\ a_2 \mid \texttt{nil} \\ p & ::= & x \mid x.sel \\ b & ::= & \texttt{true} \mid \texttt{false} \mid \texttt{not}\ b \mid b_1\ op_b\ b_2 \mid a_1\ op_r\ a_2 \mid op_p\ p \\ S & ::= & [p\texttt{:=}a]^\ell \mid [\texttt{skip}]^\ell \mid S_1;\ S_2 \mid \\ & & \texttt{if}\ [b]^\ell\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid \texttt{while}\ [b]^\ell\ \texttt{do}\ S \mid \\ & & [\texttt{malloc}\ p]^\ell \end{array}$$

Example

$[\texttt{y:=nil}]^1$;
$\texttt{while}\ [\texttt{not is-nil(x)}]^2\ \texttt{do}$
  $([\texttt{z:=y}]^3; [\texttt{y:=x}]^4; [\texttt{x:=x.cdr}]^5; [\texttt{y.cdr:=z}]^6)$;
$[\texttt{z:=nil}]^7$

Reversal of a list



Structural Operational Semantics

A configurations consists of

- a state $\sigma \in \mathbf{State} = \mathbf{Var}_\star \to (\mathbf{Z} + \mathbf{Loc} + \{\diamond\})$

  mapping variables to values, locations (in the heap) or the nil-value

- a heap $\mathcal{H} \in \mathbf{Heap} = (\mathbf{Loc} \times \mathbf{Sel}) \to_{\text{fin}} (\mathbf{Z} + \mathbf{Loc} + \{\diamond\})$

  mapping pairs of locations and selectors to values, locations in the heap or the nil-value

Pointer expressions

$$\wp : \mathbf{PExp} \rightarrow (\mathbf{State} \times \mathbf{Heap}) \rightarrow_{\text{fin}} (\mathbf{Z} + \{\diamond\} + \mathbf{Loc})$$

is defined by

$$\wp[\![x]\!](\sigma, \mathcal{H}) = \sigma(x)$$

$$\wp[\![x.sel]\!](\sigma, \mathcal{H}) = \begin{cases} \mathcal{H}(\sigma(x), sel) \\ \qquad \text{if } \sigma(x) \in \mathbf{Loc} \text{ and } \mathcal{H} \text{ is defined on } (\sigma(x), sel) \\ \text{undefined} \qquad \text{otherwise} \end{cases}$$

Arithmetic and boolean expressions

$$\mathcal{A} : \mathbf{AExp} \rightarrow (\mathbf{State} \times \mathbf{Heap}) \rightarrow_{\text{fin}} (\mathbf{Z} + \mathbf{Loc} + \{\diamond\})$$

$$\mathcal{B} : \mathbf{BExp} \rightarrow (\mathbf{State} \times \mathbf{Heap}) \rightarrow_{\text{fin}} \mathbf{T}$$

Statements

Clauses for assignments:

$$\langle [x := a]^\ell, \sigma, \mathcal{H} \rangle \rightarrow \langle \sigma[x \mapsto \mathcal{A}[\![a]\!](\sigma, \mathcal{H})], \mathcal{H} \rangle$$
if $\mathcal{A}[\![a]\!](\sigma, \mathcal{H})$ is defined

$$\langle [x.sel := a]^\ell, \sigma, \mathcal{H} \rangle \rightarrow \langle \sigma, \mathcal{H}[(\sigma(x), sel) \mapsto \mathcal{A}[\![a]\!](\sigma, \mathcal{H})] \rangle$$
if $\sigma(x) \in \mathbf{Loc}$ and $\mathcal{A}[\![a]\!](\sigma, \mathcal{H})$ is defined

Clauses for malloc:

$$\langle [\texttt{malloc } x]^\ell, \sigma, \mathcal{H} \rangle \rightarrow \langle \sigma[x \mapsto \xi], \mathcal{H} \rangle$$
where ξ does not occur in σ or $\mathcal{H}$

$$\langle [\texttt{malloc } (x.sel)]^\ell, \sigma, \mathcal{H} \rangle \rightarrow \langle \sigma, \mathcal{H}[(\sigma(x), sel) \mapsto \xi] \rangle$$
where ξ does not occur in σ or $\mathcal{H}$ and $\sigma(x) \in \mathbf{Loc}$

Shape graphs

The analysis will operate on *shape graphs* (S, H, is) consisting of

- an abstract state, S,
- an abstract heap, H, and
- sharing information, is, for the abstract locations.

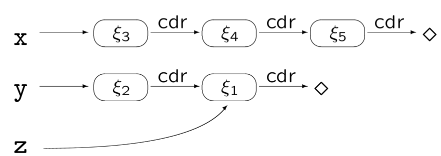
The nodes of the shape graphs are abstract locations:

$$\mathbf{ALoc} = \{n_X \mid X \subseteq \mathbf{Var}_\star\}$$

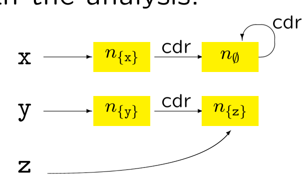
Note: there will only be *finitely* many abstract locations

Example

In the semantics:



In the analysis:



Abstract Locations

The abstract location n_X represents the location $\sigma(x)$ if $x \in X$

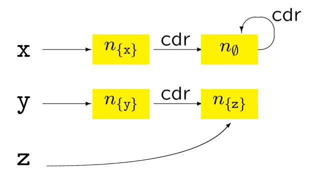
The abstract location $n_\emptyset$ is called the *abstract summary location*: $n_\emptyset$ represents all the locations that cannot be reached directly from the state without consulting the heap

Invariant 1 If two abstract locations n_X and n_Y occur in the same shape graph then either $X = Y$ or $X \cap Y = \emptyset$

Abstract states and heaps

$$S \in \mathbf{AState} = \mathcal{P}(\mathbf{Var}_\star \times \mathbf{ALoc}) \quad \text{abstract states}$$

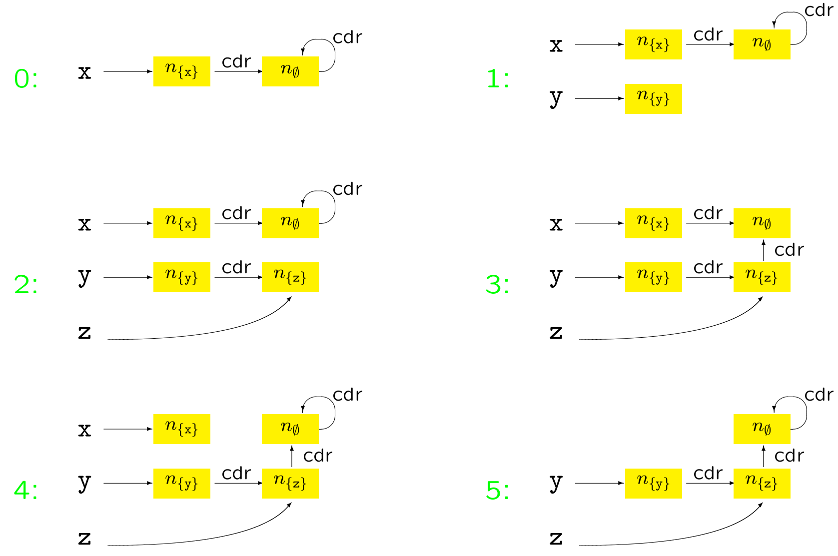
$$H \in \mathbf{AHeap} = \mathcal{P}(\mathbf{ALoc} \times \mathbf{Sel} \times \mathbf{ALoc}) \quad \text{abstract heap}$$



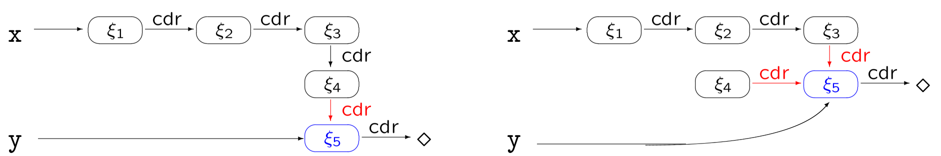
Invariant 2 If x is mapped to n_X by the abstract state S then $x \in X$

Invariant 3 Whenever (n_V, sel, n_W) and $(n_V, sel, n_{W'})$ are in the abstract heap H then either $V = \emptyset$ or $W = W'$

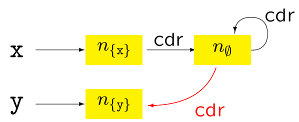
Reversal of a list



Sharing in the heap



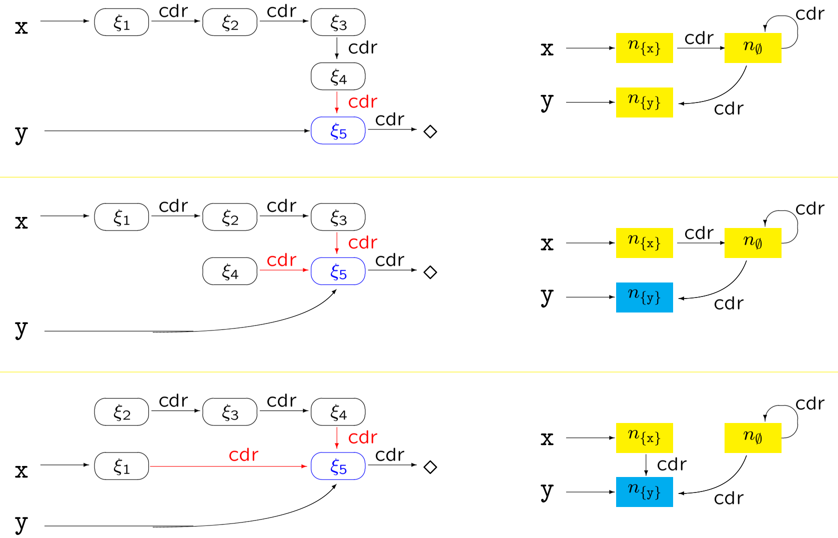
Give rise to the same shape graph:



is: the abstract locations that *might* be shared due to pointers in the heap:

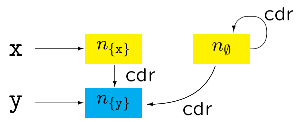
n_X is included in **is** if it might represents a location that is the target of more than one pointer in the heap

Examples: sharing in the heap



Sharing information

The implicit sharing information of the abstract heap must be consistent with the explicit sharing information:



Invariant 4 If $n_X \in$ is then either

- $(n_\emptyset, sel, n_X)$ is in the abstract heap for some *sel*, or
- there are two distinct triples (n_V, sel_1, n_X) and (n_W, sel_2, n_X) in the abstract heap

Invariant 5 Whenever there are two distinct triples (n_V, sel_1, n_X) and (n_W, sel_2, n_X) in the abstract heap and $X \neq \emptyset$ then $n_X \in$ is

The complete lattice of shape graphs

A *shape graph* is a triple (S,H,is) where

$$\begin{array}{lcl} \mathsf{S} \in \mathbf{AState} &=& \mathcal{P}(\mathbf{Var}_\star \times \mathbf{ALoc}) \\ \mathsf{H} \in \mathbf{AHeap} &=& \mathcal{P}(\mathbf{ALoc} \times \mathbf{Sel} \times \mathbf{ALoc}) \\ \mathsf{is} \in \mathbf{IsShared} &=& \mathcal{P}(\mathbf{ALoc}) \end{array}$$

and $\mathbf{ALoc} = \{n_Z \mid Z \subseteq \mathbf{Var}_\star\}$.

A shape graph (S, H, is) is *compatible* if it fulfils the five invariants.

The analysis computes over *sets of compatible shape graphs*

$$\mathbf{SG} = \{(\mathsf{S}, \mathsf{H}, \mathsf{is}) \mid (\mathsf{S}, \mathsf{H}, \mathsf{is}) \text{ is compatible}\}$$

The analysis

An instance of a *forward* Monotone Framework with the complete lattice of interest being $\mathcal{P}(\mathbf{SG})$

A *may analysis*: each of the sets of shape graphs computed by the analysis may contain shape graphs that cannot really arrise

Aspects of a *must analysis*: each of the individual shape graphs (in a set of shape graphs computed by the analysis) will be the best possible description of some $(\sigma, \mathcal{H})$

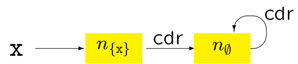
The analysis

Equations:

$$Shape_\circ(\ell) = \begin{cases} \iota & \text{if } \ell = init(S_\star) \\ \bigcup\{Shape_\bullet(\ell') \mid (\ell', \ell) \in flow(S_\star)\} & \text{otherwise} \end{cases}$$

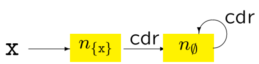
$$Shape_\bullet(\ell) = f_\ell^{\mathsf{SA}}(Shape_\circ(\ell))$$

Example: The extremal value ι for the list reversal program



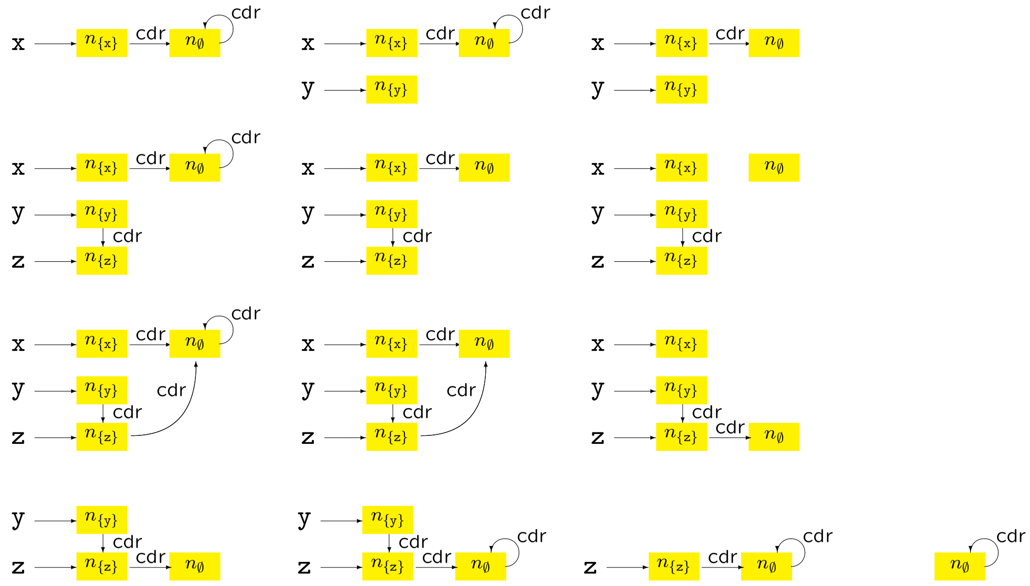
– x points to a non-cyclic list with at least three elements

Shape•(1) for [y:=nil]^1

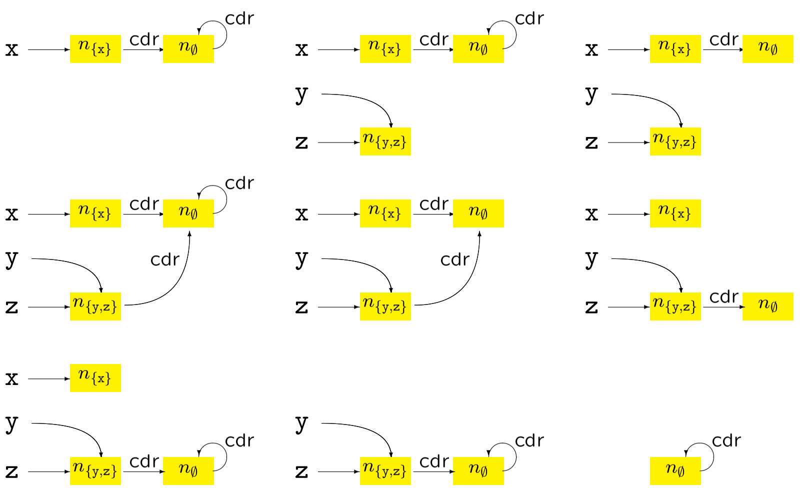


Note: we do not record `nil`-values in the analysis

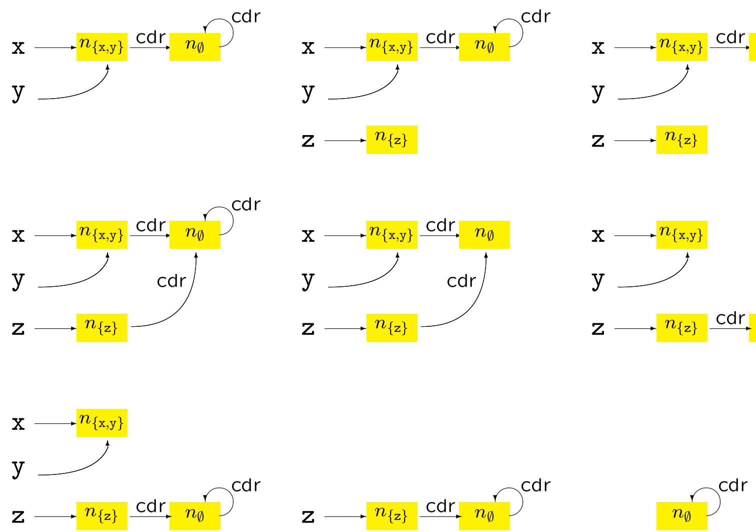
Shape•(2) for [not is-nil(x)]^2



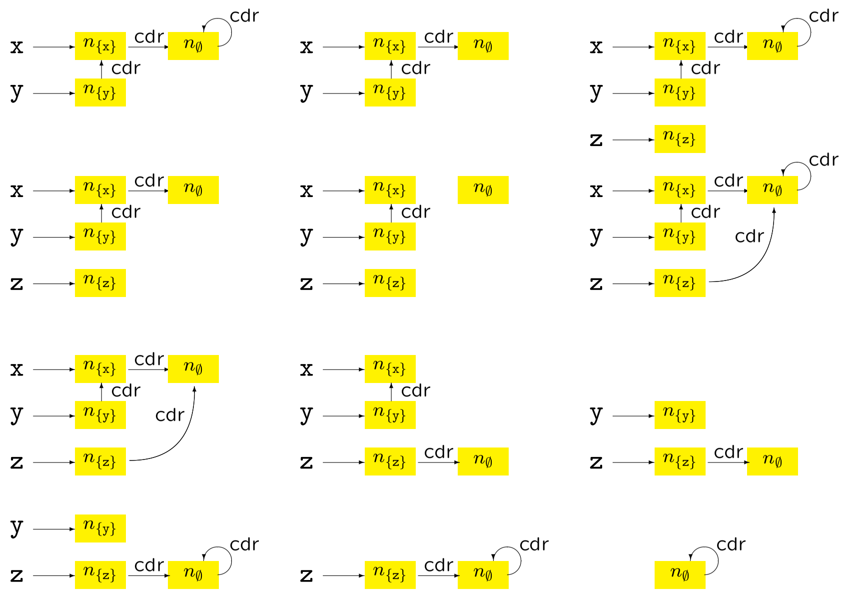
Shape•(3) for [z:=y]^3



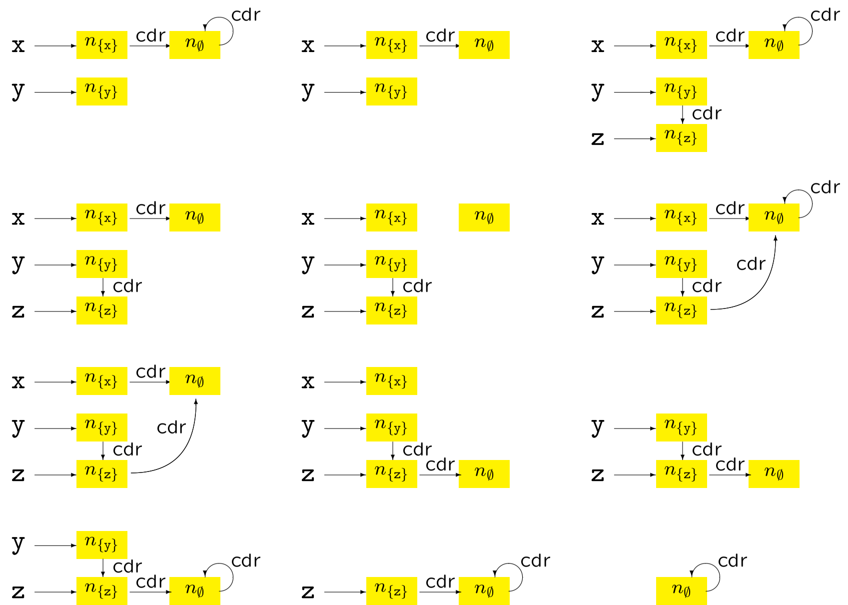
Shape•(4) for [y:=x]^4



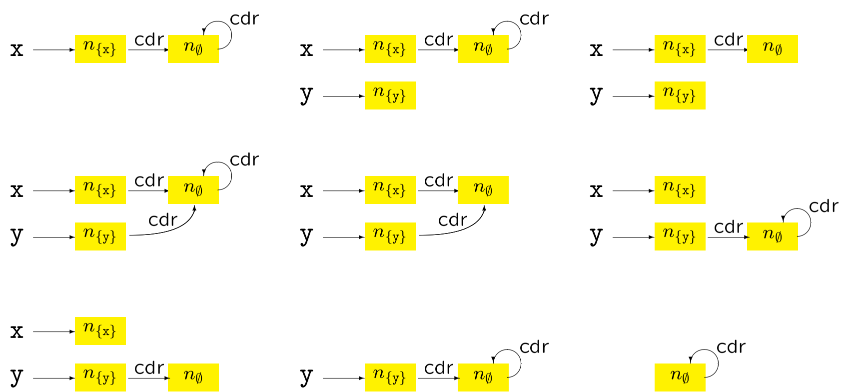
Shape$_\bullet$(5) for [x:=x.cdr]^5



Shape$_\bullet$(6) for [y.cdr:=z]^6



Shape$_\bullet$(7) for [z:=nil]^7



– upon termination `y` points to a non-circular list

– a more precise analysis taking tests into account will know that `x` is `nil` upon termination

Transfer functions

$$f_\ell^{\mathsf{SA}} : \mathcal{P}(\mathbf{SG}) \to \mathcal{P}(\mathbf{SG})$$

has the form:

$$f_\ell^{\mathsf{SA}}(SG) = \bigcup\{\phi_\ell^{\mathsf{SA}}((\mathsf{S},\mathsf{H},\mathsf{is})) \mid (\mathsf{S},\mathsf{H},\mathsf{is}) \in SG\}$$

where

$$\phi_\ell^{\mathsf{SA}} : \mathbf{SG} \to \mathcal{P}(\mathbf{SG})$$

specifies how a *single* shape graph (in *Shape*$_\circ(\ell)$) may be transformed into a *set* of shape graphs (in *Shape*$_\bullet(\ell)$) by the elementary block.

Transfer function for $[b]^\ell$ and $[\texttt{skip}]^\ell$

We are only interested in the shape of the heap – and it is not changed by these elementary blocks:

$$\phi_\ell^{\mathsf{SA}}((\mathsf{S},\mathsf{H},\mathsf{is})) = \{(\mathsf{S},\mathsf{H},\mathsf{is})\}$$

Transfer function for $[x := a]^\ell$

— where a is of the form n, a_1 op_a a_2 or `nil`

$$\phi_\ell^{\mathsf{SA}}((\mathsf{S},\mathsf{H},\mathsf{is})) = \{kill_x((\mathsf{S},\mathsf{H},\mathsf{is}))\}$$

where $kill_x((\mathsf{S},\mathsf{H},\mathsf{is})) = (\mathsf{S}',\mathsf{H}',\mathsf{is}')$ is

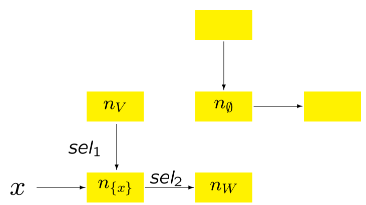
$$\begin{array}{rcl} \mathsf{S}' &=& \{(z, k_x(n_Z)) \mid (z, n_Z) \in S \ \wedge \ z \neq x\} \\ \mathsf{H}' &=& \{(k_x(n_V), sel, k_x(n_W)) \mid (n_V, sel, n_W) \in \mathsf{H}\} \\ \mathsf{is}' &=& \{k_x(n_X) \mid n_X \in \mathsf{is}\} \end{array}$$

and

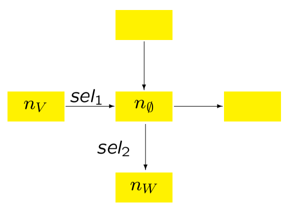
$$k_x(n_Z) = n_{Z \setminus \{x\}}$$

Idea: all abstract locations are renamed to not having x in their name set

The effect of $[x := \texttt{nil}]^\ell$



$(\mathsf{S},\mathsf{H},\mathsf{is})$



$(\mathsf{S}',\mathsf{H}',\mathsf{is}')$

Transfer function for $[x := y]^\ell$ when $x \neq y$

$$\phi_\ell^{\mathsf{SA}}((\mathsf{S},\mathsf{H},\mathsf{is})) = \{(\mathsf{S}'',\mathsf{H}'',\mathsf{is}'')\}$$

where $(\mathsf{S}',\mathsf{H}',\mathsf{is}') = kill_x((\mathsf{S},\mathsf{H},\mathsf{is}))$ and

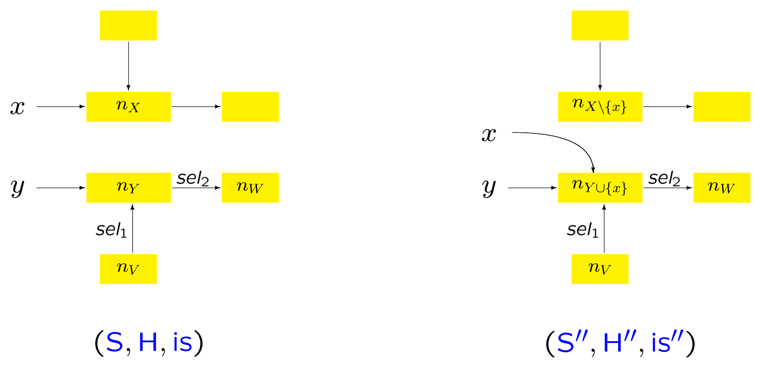
$$\begin{array}{rcl} \mathsf{S}'' &=& \{(z, g_x^y(n_Z)) \mid (z, n_Z) \in \mathsf{S}'\} \\ && \cup \ \{(x, g_x^y(n_Y)) \mid (y', n_Y) \in \mathsf{S}' \wedge y' = y\} \\ \mathsf{H}'' &=& \{(g_x^y(n_V), sel, g_x^y(n_W)) \mid (n_V, sel, n_W) \in \mathsf{H}'\} \\ \mathsf{is}'' &=& \{g_x^y(n_Z) \mid n_Z \in \mathsf{is}'\} \end{array}$$

and

$$g_x^y(n_Z) = \begin{cases} n_{Z \cup \{x\}} & \text{if } y \in Z \\ n_Z & \text{otherwise} \end{cases}$$

Idea: all abstract locations are renamed to also have x in their name set if they already have y

The effect of $[x := y]^\ell$ when $x \neq y$



Transfer function for $[x := y.sel]^\ell$ when $x \neq y$

Remove the old binding for x: strong nullification

$$(S', H', is') = kill_x((S, H, is))$$

Establish the new binding for x:

1. There is no abstract location n_Y such that $(y, n_Y) \in S'$ – or there is an abstract location n_Y such that $(y, n_Y) \in S'$ but no n_Z such that $(n_Y, sel, n_Z) \in H'$
2. There is an abstract location n_Y such that $(y, n_Y) \in S'$ and there is an abstract location $n_U \neq n_\emptyset$ such that $(n_Y, sel, n_U) \in H'$
3. There is an abstract location n_Y such that $(y, n_Y) \in S'$ and $(n_Y, sel, n_\emptyset) \in H'$

Case 1 for $[x := y.sel]^\ell$

Assume there is no abstract location n_Y such that $(y, n_Y) \in S'$

$$\phi_\ell^{SA}((S, H, is)) = \{(S', H', is')\}$$

OBS: dereference of a `nil`-pointer

Assume there is an abstract location n_Y such that $(y, n_Y) \in S'$ but there is no abstract location n such that $(n_Y, sel, n) \in H'$

$$\phi_\ell^{SA}((S, H, is)) = \{(S', H', is')\}$$

OBS: dereference of a non-existing *sel*-field

Case 2 for $[x := y.sel]^\ell$

Assume there is an abstract location n_Y such that $(y, n_Y) \in S'$ and there is an abstract location $n_U \neq n_\emptyset$ such that $(n_Y, sel, n_U) \in H'$.

The abstract location n_U will be renamed to include the variable x using the function:

$$h_x^U(n_Z) = \begin{cases} n_{U\cup\{x\}} & \text{if } Z = U \\ n_Z & \text{otherwise} \end{cases}$$

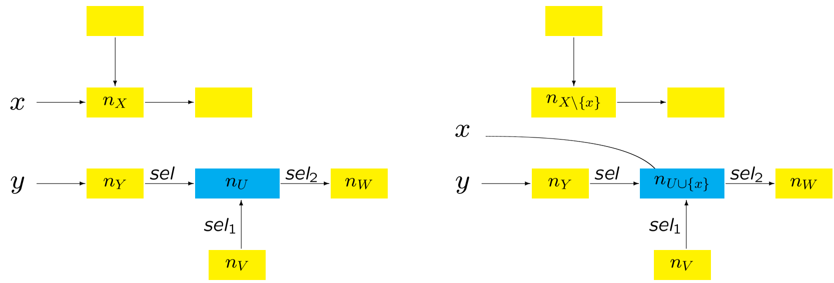
We take

$$\phi_\ell^{SA}((S, H, is)) = \{(S'', H'', is'')\}$$

where $(S', H', is') = kill_x((S, H, is))$ and

$$\begin{aligned} S'' &= \{(z, h_x^U(n_Z)) \mid (z, n_Z) \in S'\} \cup \{(x, h_x^U(n_U))\} \\ H'' &= \{(h_x^U(n_V), sel', h_x^U(n_W)) \mid (n_V, sel', n_W) \in H'\} \\ is'' &= \{h_x^U(n_Z) \mid n_Z \in is'\} \end{aligned}$$

The effect of $[x := y.sel]^\ell$ in Case 2



(S, H, is) (S″, H″, is″)

Case 3 for $[x := y.sel]^\ell$ (1)

Assume that there is an abstract location n_Y such that $(y, n_Y) \in S'$ and furthermore $(n_Y, sel, n_\emptyset) \in H'$.

We have to *materialise* a new abstract location $n_{\{x\}}$ from $n_\emptyset$.

$[x:=\texttt{nil}]^{\cdots}$; $[x:=y.sel]^\ell$; $[x:=\texttt{nil}]^{\cdots}$

(S, H, is) (S′, H′, is′) (S″, H″, is″) (S‴, H‴, is‴)

Idea:

$$(S', H', is') = (S''', H''', is''') = kill_x((S'', H'', is''))$$

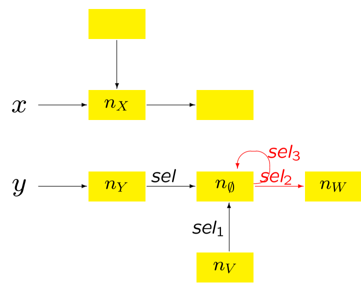
Case 3 for $[x := y.sel]^\ell$ (2)

Transfer function:

$$\phi_\ell^{SA}((S,H,is)) = \{(S'',H'',is'') \mid (S'',H'',is'') \text{ is compatible} \wedge kill_x((S'',H'',is'')) = (S',H',is') \wedge (x, n_{\{x\}}) \in S'' \wedge (n_Y, sel, n_{\{x\}}) \in H''\}$$

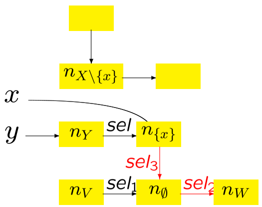
where $(S', H', is') = kill_x((S, H, is))$.

The effect of $[x := y.sel]^\ell$ in Case 3 (1)

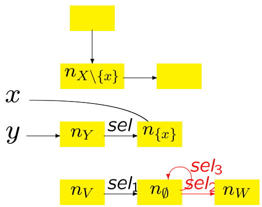


(S, H, is)

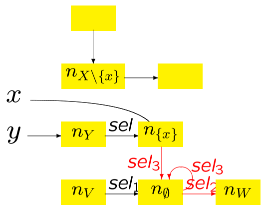
The effect of $[x{:=}y.sel]^\ell$ in Case 3 (2)



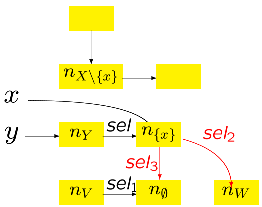
(S''_1, H''_1, is''_1)



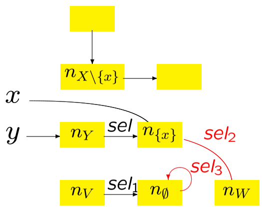
(S''_3, H''_3, is''_3)



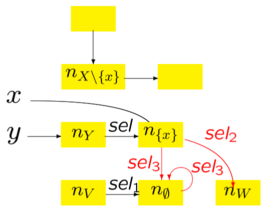
(S''_5, H''_5, is''_5)



(S''_2, H''_2, is''_2)



(S''_4, H''_4, is''_4)



(S''_6, H''_6, is''_6)

Transfer function for $[x.sel{:=}a]^\ell$

— where a is of the form n, a_1 op_a a_2 or `nil`.

If there is no n_X such that $(x, n_X) \in S$ then f_ℓ^{SA} is the identity.

If there is n_X such that $(x, n_X) \in S$ but that there is no n_U such that $(n_X, sel, n_U) \in H$ then f_ℓ^{SA} is the identity.

If there are abstract locations n_X and n_U such that $(x, n_X) \in S$ and $(n_X, sel, n_U) \in H$ then

$$\phi_\ell^{SA}((S, H, is)) = \{kill_{x.sel}((S, H, is))\}$$

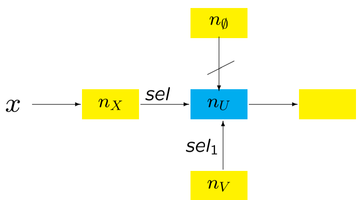
where $kill_{x.sel}((S, H, is)) = (S', H', is')$ is given by

$$S' = S$$

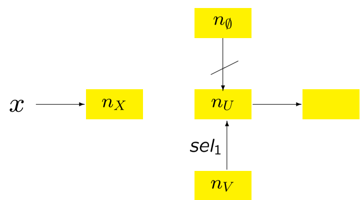
$$H' = \{(n_V, sel', n_W) \mid (n_V, sel', n_W) \in H \wedge \neg(X = V \wedge sel = sel')\}$$

$$is' = \begin{cases} is \backslash \{n_U\} & \text{if } n_U \in is \wedge \#into(n_U, H') \leq 1 \wedge \neg\exists(n_\emptyset, sel', n_U) \in H' \\ is & \text{otherwise} \end{cases}$$

The effect of $[x.sel{:=}\texttt{nil}]^\ell$ when $\#into(n_U, H') \leq 1$



(S, H, is)



(S', H', is')

Transfer function for $[x.sel{:=}y]^\ell$ when $x \neq y$

If there is no n_X such that $(x, n_X) \in S$ then f_ℓ^{SA} is the identity function.

If $(x, n_X) \in S$ but there is no n_Y such that $(y, n_Y) \in S$ then

$$\phi_\ell^{SA}((S, H, is)) = \{kill_{x.sel}((S, H, is))\}$$

If there is $(x, n_X) \in S$ and $(y, n_Y) \in S$ then

$$\phi_\ell^{SA}((S, H, is)) = \{(S'', H'', is'')\}$$

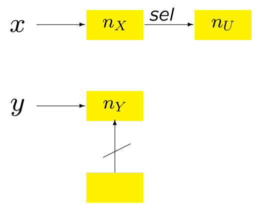
where $(S', H', is') = kill_{x.sel}((S, H, is))$ and

$$S'' = S' \quad (= S)$$

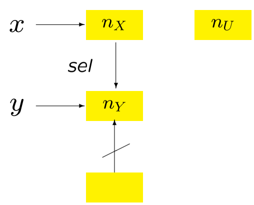
$$H'' = H' \cup \{(n_X, sel, n_Y) \mid (x, n_X) \in S' \wedge (y, n_Y) \in S'\}$$

$$is'' = \begin{cases} is' \cup \{n_Y\} & \text{if } \#into(n_Y, H') \geq 1 \\ is' & \text{otherwise} \end{cases}$$

The effect of $[x.sel{:=}y]^\ell$ when $\#into(n_Y, \mathsf{H}') \leq 1$



$(\mathsf{S}, \mathsf{H}, \mathsf{is})$



$(\mathsf{S}', \mathsf{H}'', \mathsf{is}'')$

Transfer function for $[\texttt{malloc}\ x]^\ell$

$$\phi_\ell^{\mathsf{SA}}((\mathsf{S}, \mathsf{H}, \mathsf{is})) = \{(\mathsf{S}' \cup \{(x, n_{\{x\}})\}, \mathsf{H}', \mathsf{is}')\}$$

where $(\mathsf{S}', \mathsf{H}', \mathsf{is}') = kill_x(\mathsf{S}, \mathsf{H}, \mathsf{is})$.