

## Shapes/Slotted E-graphs

### Definition slotted e-graph

Similar to (regular) egraphs:

- function symbols  $f, g$
- e-class ids  $a, b, c$
- slots  $s_1, s_2, \dots$
- slotmap  $m ::= [s_j \mapsto s_k, \dots]$  bijection
- invocation  $i ::= m * a$
- terms  $t ::= f \mid f(t_1, \dots, t_k) \mid s_j \mid \lambda s_j. t$
- e-nodes  $n ::= f \mid f(i_1, \dots, i_k) \mid s_j \mid \lambda s_j. i$
- e-classes  $c ::= \{n_1, \dots, n_m\} :: \{s_{i_1}, \dots, s_{i_k}\}$

We have a mapping  $\text{Classes} : \text{Id} \rightarrow \text{Eclass}$  to interpret the invocations.

### Additional definitions

#### slots(—)

We define a family of (overloaded) functions  $\text{slots}$  for e-class ids, invocations, terms and e-nodes to a set of slots  $\{s_{i_1}, \dots, s_{i_k}\}$ .

#### slots(—) for E-Classes, Ids and Invocations

- $\text{slots}(\{n_1, \dots, n_m\} :: \{s_{i_1}, \dots, s_{i_k}\}) := \{s_{i_1}, \dots, s_{i_k}\}$
- $\text{slots}(a) := \text{slots}(\text{Classes}(a))$
- $\text{slots}(m * a) := m \circ \text{slots}(a) = \{m(s_j) \mid s_j \in \text{slots}(a)\}$

**slots(—) for Terms and E-Nodes** Let  $x, x_1, \dots$  be either terms  $t$  or invocations  $i$ .

- $\text{slots}(f) := \emptyset$
- $\text{slots}(f(x_1, \dots, x_k)) := \text{slots}(x_1) \cup \dots \cup \text{slots}(x_k)$
- $\text{slots}(s_j) := \{s_j\}$
- $\text{slots}(\lambda s_j. x) := \text{slots}(x) \setminus \{s_j\}$

$\text{slots}(t)$  on terms corresponds to the set of free variables.

#### The Action $m * \_$

We define a family of (overloaded) functions  $m * \_$  for e-class ids, invocations, terms and e-nodes.

Generally,  $m * x$  is only defined, if  $\text{slots}(x) \subseteq \text{dom}(m)$ .

- For any  $x$  we have  $m * (m' * x) = (m' * m) * x$
- with  $m * m' := m \circ m' = \{x \mapsto z \mid x \mapsto y \in m', y \mapsto z \in m\}$

### **m \* \_\_ for E-Classes, Ids and Invocations**

- $m * \{n_1, \dots, n_k\} :: \{s_{i_1}, \dots, s_{i_l}\} = \{m * n_1, \dots, m * n_k\} :: \{m * s_{i_1}, \dots, m * s_{i_l}\}$
- $m * a$  is just  $m * a$ , there is no way to simplify it
- For Invocations  $i = m * a$ , we define  $m' * i := (m' * m) * a$

Note that there is a difference in semantics between e-classes and e-class ids for this action. So it might be necessary to keep them apart.

### **m \* \_\_ for Terms and E-Nodes** Let $x, x_1, \dots$ be either terms $t$ or invocations $i$ .

- $m * f := f$
- $m * f(x_1, \dots, x_k) := f(m * x_1, \dots, m * x_k)$
- $m * s_j := m(s_j)$
- $m * (\lambda s_j. x) := \lambda s_j. (m * x)$ , assuming  $s_j$  is neither in the domain nor codomain of  $m$ .

We follow the Barendregt convention: We assume that all bound slots are never colliding with anything else. And if they do, we just rename them.

(Note to future self: We also need the Barendregt convention for redundant slots)

We claim that this definition implies  $\text{slots}(m * x) = m \circ \text{slots}(x)$  for all  $x$ .

### **Examples:**

- $\text{slots}(\lambda s_1. f(s_1, s_2, s_3)) = \text{slots}(f(s_1, s_2, s_3)) \setminus \{s_1\} = \{s_2, s_3\}$
- $\lambda s_1. f(s_1, s_2, s_3) * (s_1 \mapsto s_2, s_2 \mapsto s_3, s_3 \mapsto s_1)$  does not typecheck
- $\lambda s_1. f(s_1, s_2, s_3) * (s_2 \mapsto s_3, s_3 \mapsto s_1) = \lambda s_1. f(s_1, s_2, s_3)$  needs freshness
- $[s_2 \mapsto s_3, s_{47} \mapsto s_2] * a = [s_{47} \mapsto s_2, s_2 \mapsto s_3] * a; \text{slots}(a) = \{s_2, s_{47}\}$

## **Containment**

We define an element relation  $\in \subseteq \text{Enodes} \times \text{Invocations}$  as: -  $n \in m * a$ , iff  $n \in m * \text{Classes}(a)$

### **Notes**

- We need to be more precise about the slots of e-nodes and e-classes (union/intersection, etc)
- We allow the shortcut for ordered (instead of named) arguments,  $[s_j, s_{j'}, s_{j''}, \dots] * i := [s_k \mapsto s_j, s_{k'} \mapsto s_{j'}, s_{k''} \mapsto s_{j''}, \dots] * i$ , assuming  $\text{slots}(i) = \{s_k, s_{k'}, s_{k''}, \dots\}$  and  $k < k' < k'' < \dots$

## Automorphism Group

The operator  $*$  defines a left group action of the group  $G \leq \text{Sym}(\text{slots}(x))$ .

The automorphism group  $\text{Aut}(c)$  of an e-class  $c = \{n_1, \dots, n_m\} :: \{s_{i_1}, \dots, s_{i_k}\}$  is the largest subgroup  $\text{Aut}(c) \leq \text{Sym}(s_{i_1}, \dots, s_{i_m})$ , such that  $m * c = c$  for all  $m \in \text{Aut}(c)$ .

## Orbits, Canonical Elements

For an e-node  $n$  and a group of slotmaps  $M \leq \text{slots}(n)$ , the orbit  $M * n$  is the set of all permutations of  $n$  according to the group  $M$ , i.e.  $M * n = \{m * n \mid m \in M\}$ . Given a term ordering (we assume lexicographical)  $<$ , we define a canonical element of the orbit  $M * n := \min_{m \in M} m * n$  to be the minimal representative of the orbit.

## Weak Shapes

We define the weak shape of an e-node  $n$  as follows:

- $\text{weak\_shape}(n) := \min\{\text{Sym}(S) * n\}$ , where  $S = \{s_j \mid j \in \mathbb{N}\}$  is the set of all slots.

## Example

- Consider the e-class  $c = s_0 + s_1, s_1 + s_0 :: s_0, s_1$ , then the two e-nodes  $f([s_2, s_3] * c, [s_2, s_3] * c)$  and  $f([s_2, s_3] * c, [s_3, s_2] * c)$  should have the same hash because  $[s_2, s_3] * c = [s_3, s_2] * c$ . However, they don't have the same weak shape, because we don't compute the (weak shapes) of the invocations  $[s_2, s_3] * c$ .

## Strong shape

- $\text{strong\_shape}(f(m_1 * c_1, \dots, m_k * c_k)) = \min\{\text{weak\_shape}(f(m_1 * m'_1 * c_1, \dots, m_k * m'_k * c_k)) \mid m'_i \text{ in } \text{Aut}(c_i)\}$
- This typechecks because  $\text{slots}(m * c) = \text{slots}(c)$  for all  $m \in \text{Aut}(c)$
- enodes must be hashable
- hash must be invariant of renamings
- weak shape: canonical naming  $s_1, s_2, \dots$
- egraph idea: congruence, i.e. if  $a = b \Rightarrow f(a) = f(b)$  (in memory). This does not work in weak shapes:
- example:  $\text{Classes}(a) = \{(f s_1 0 s_1 1), (f s_1 1 s_1 0)\} :: \{s_1 0, s_1 1\}$  enodes:  $(+[s_1 0 \rightarrow s_1, s_1 1 \rightarrow s_2] * a [s_1 0 \rightarrow s_2, s_1 1 \rightarrow s_1] * a)$  and  $(+[s_1 0 \rightarrow s_1, s_1 1 \rightarrow s_2] * a [s_1 0 \rightarrow s_1, s_1 1 \rightarrow s_2] * a)$  concrete terms correspond to  $\$(+ (f \times y) (f y \times))\$$  and  $+(fxy)(fxy)$
- strong shape: lex-min of all equivalent weak shapes.
- Conjecture: this is the double coset constructive orbit problem

## Open questions

- Is  $\lambda$  above the most generic possible, or are there examples of languages where the binders cannot be expressed this way?