Lab Week 9—Efficient Records in rePL

CS 4215: Programming Language Implementation

Martin Henz

March 13, 2012

Generated on Thursday 15 March, 2012, 22:43
New Instructions for Record Construction

New instructions:

\[
\begin{align*}
\text{LDPS } q.s & \\
\text{RCDS } i.s & \\
\end{align*}
\]

where \( q \) is a property and \( i \) is an integer.
Compilation of Record Construction

\[ E_1 \leftrightarrow s_1 \quad \ldots \quad E_n \leftrightarrow s_n \]

\[ [q_1 : E_1, \ldots, q_n : E_n] \leftrightarrow \text{LDPS } q_1.s_1 \ldots \text{LDPS } q_n.s_n.\text{RCDS } n \]
Execution of LDPS

\[ s(pc) = \text{LDPS } q \]

\[ (os, pc, e, rs) \Rightarrow_s (q.os, pc + 1, e, rs) \]
Consider LDPS $q_1.s_1$\ldots LDPS $q_n.s_n$. RCDS $n$ resulting from $[q_1:E_1, \ldots, q_n:E_n]$.

After LDPS $q_1.s_1$\ldots LDPS $q_n.s_n$, instruction RCDS $n$ finds association list on operand stack.

$$s(pc) = \text{RCDS } n$$

$$\left( v_n.q_n.\ldots.v_1.q_1.os, pc, e, rs \right) \xrightarrow{s} \left( \{(q_1, v_1), \ldots, (q_n, v_n)\}.os, pc + 1, e, rs \right)$$
Operations on Records

Operations: empty, ".", hasproperty
New instructions:

\[
\begin{align*}
\text{EMPTY}.s & \quad s & \quad \text{DOT}.s & \quad s & \quad \text{HASP}.s
\end{align*}
\]
Compilation of Record Operations

\[ E \leftarrow s \]

\[ \text{empty } E \leftarrow s.\text{EMPTY} \]

\[ E \leftarrow s \]

\[ E \text{ has property } q \leftarrow s.\text{LDPS } q.\text{DOT} \]

\[ E \text{ has property } q \leftarrow s.\text{LDPS } q.\text{HASP} \]
Executions of Record Operations

\[
s(pc) = \text{EMPTY}
\]

if \( v = \emptyset \)

\[
(v.os, pc, e, rs) \xrightarrow{s} (true.os, pc + 1, e, rs)
\]

\[
s(pc) = \text{EMPTY}
\]

if \( v \neq \emptyset \)

\[
(v.os, pc, e, rs) \xrightarrow{s} (false.os, pc + 1, e, rs)
\]
Executions of Record Operations

\[
\begin{align*}
    s(pc) &= \text{DOT} \\
    \quad &\text{if } \nu(q) = \nu' \\
    (q.v.os, pc, e, rs) &\Rightarrow_s (\nu'.os, pc + 1, e, rs)
\end{align*}
\]

\[
\begin{align*}
    s(pc) &= \text{HASP} \\
    \quad &\text{if } \exists v_i.(q, v_i) \in v \\
    (q.v.os, pc, e, rs) &\Rightarrow_s (true.os, pc + 1, e, rs)
\end{align*}
\]

\[
\begin{align*}
    s(pc) &= \text{HASP} \\
    \quad &\text{if } \forall v_i.(q, v_i) \in v \\
    (q.v.os, pc, e, rs) &\Rightarrow_s (false.os, pc + 1, e, rs)
\end{align*}
\]
Division by zero and record access throw exceptions. Idea: place instructions for raising these two exceptions at the end of the instruction sequence.

\[ E \rightarrow s_1 \]

\[ \text{[divisionByZero: true]} \rightarrow s_2 \]

\[ \text{[invalidRecordAccess: true]} \rightarrow s_3 \]

\[ Es_1 . \text{DONE} . s_2 . \text{THROW} . s_3 . \text{THROW} \]

beginning address of \( s_2 \): \( addr_{\text{divisionByZero}} \)

beginning address of \( s_3 \): \( addr_{\text{invalidRecordAccess}} \)
Primitive Operations Throwing Exceptions

\[ s(pc) = \text{DIV} \]

\[ (0.1.os, pc, e, rs) \xrightarrow{s} (os, addr_{\text{divisionByZero}}, e, rs) \]

\[ s(pc) = \text{DOT} \]

\[ (q.v.os, pc, e, rs) \xrightarrow{s} (os, addr_{\text{invalidRecordAccess}}, e, rs) \]

if \( \forall v.(q, v') \in v \)
Programmer-defined Exception Throws

\[
E \xrightarrow{s}
\]

\[
\text{throw } E \text{ end } \xrightarrow{s}.\text{THROW}
\]
Use runtime stack to keep track of the catch...with... part of try expressions

Exception from try part will pop stackframes, until it finds the appropriate catch...with... part
Translation of \texttt{try} Statement

\begin{align*}
E_1 & \rightarrow s_1 \quad E_2 \rightarrow s_2 \\
\text{try } E_1 \text{ catch } x \text{ with } E_2 \text{ end} & \rightarrow (\text{TRY } x |s_1| + 3).s_1.\text{ENDTRY.}(\text{GOTOR } s_2| + 1).s_2
\end{align*}
Execution of TRY Instruction

\[ s(pc) = \text{TRY} \times i \]

\[
(os, pc, e, rs) \stackrel{s}{\Rightarrow} (os, pc + 1, e, (\text{catch}, x, pc + i, os, e).rs)
\]
Execution of ENDTRY Instruction

\[ s(pc) = \text{ENDTRY} \]

\[ (os, pc, e, (\text{catch}, x, pc', os, e).rs) \xrightarrow{s} (os, pc + 1, e, rs) \]
Throwing of an Exception

\[ s(pc) = \text{THROW} \]

\[ (os, pc, e, (pc', os', e').rs) \xrightarrow{s} (os, pc, e, rs) \]

\[ s(pc) = \text{THROW} \]

\[ (v.os, pc, e, (\text{catch}, x, pc', os', e').rs) \xrightarrow{s} (os', pc', e'[x \leftarrow v], rs) \]
Problems with Records in RVM

- representation of records as a set of pairs is inefficient
- properties are strings
Observations

- Properties always appear literally
- Records are always constructed with [ . . . ], which explicitly lists all properties
Implementing Records Efficiently

Representing Properties

- Compiler constructs set $Q$ of all properties in a given $E$
- Compiler calculates a bijection $idp$ between $Q$ and $[0 \ldots |Q| - 1]$
- Compiler replaces every occurrence of a property $q$ in an instruction by $idp(q)$
Records
Exceptions
Implementing Records Efficiently

Compilation of Record Operations (revisited)

\[ E \leftrightarrow s \]

\[ E \cdot q \leftrightarrow s.LDCI \ idp(q).DOT \]

\[ E \leftrightarrow s \]

\[ E \text{ hasproperty} \ q \leftrightarrow s.LDCI \ idp(q).HASP \]
Record Construction

- All records are constructed by [...] 
- Compiler calculates a bijection $idr$ between the set $R$ of all property sets of records and $[0 \ldots |R| - 1]$. 
- Associate with each property $q$ of each record its alphabetical position in the corresponding property set: $p(idr(\{q_1, \ldots, q_n\}), idp(q))$, starting with 0. If a record with index $m$ does not have a property with index $n$, we set $p(m, n) = -1$. 
Example

- Let us say compiler assigns the number 13 to the set of properties \( \{a, b\} \) (thus \( idr(\{a, b\} = 13) \))
- the number 55 to the property \( a \)
  \( idp(a) = 55 \)
- the number 77 to the property \( b \)
  \( idp(b) = 77 \)
- the position of property \( a \) in \([a:5 \ b:7]\) is \( p(13, 55) = 0 \)
- For any property identifier \( n \neq 55, 77 \): \( p(13, n) = -1 \).
Represent a record with properties $q_1, \ldots, q_n$ as a pair consisting of identifier $idr(\{q_1, \ldots, q_n\})$ and array that maps the alphabetical position of each $q$ in the corresponding property list.
Example

In the example above, since $p(13, 55) = 0$ and $p(13, 77) = 1$, we can represent the record $[a:5 \ b:7]$ by the pair $(13, [0 : 5, 1 : 7])$. 
New Translation of Record Construction

\[ E_1 \leftrightarrow s_1 \quad \cdots \quad E_n \leftrightarrow s_n \]

\[ [q_1:E_1, \ldots, q_n:E_n] \]

\[ \mapsto \]

\[ \text{LDCI } idp(q_1).s_1 \ldots \text{LDCI } idp(q_n).s_n.\text{RCD } n \text{ } idr(\{q_1, \ldots, q_n\}) \]
Efficient Records in RVM

- compiler passes the table $p$ to RVM
- RCD constructs an array, whose indices corresponding to the record properties are given by $p$. 
New Execution of Record Construction

\[ s(pc) = RCD\ n\ m \]

\[(v_n.i_n \ldots v_1.i_1.os, pc, e, rs) \Rightarrow_s ((m, \{(p(m, idp(q_1)), v_1)\}, \ldots, (p(m, idp(q_n)), v_n)}).os, pc + 1, e, rs)\]
Implementing Records Efficiently

New Execution of Record Operations

\[ s(pc) = \text{DOT} \]

\[ (i.(m,a).os, pc, e, rs) \xrightarrow{s} (a(j).os, pc + 1, e, rs) \]

if

\[ p(m, i) = j, j \geq 0 \]

\[ s(pc) = \text{DOT} \]

\[ (i.(m,a).os, pc, e, rs) \xrightarrow{s} (os, addr_{invalidRecordAccess}, e, rs) \]

if

\[ p(m, i) = -1 \]
New Execution of Record Operations

\[ s(pc) = \text{HASP} \]

\[
\begin{array}{c}
\text{if } p(m, i) \geq 0 \\
(i.(m, a).os, pc, e, rs) \Rightarrow_s (true.os, pc + 1, e, rs)
\end{array}
\]

\[ s(pc) = \text{HASP} \]

\[
\begin{array}{c}
\text{if } (i.(m, a).os, pc, e, rs) \Rightarrow_s (false.os, pc + 1, e, rs) \\
p(m, i) = -1
\end{array}
\]
Summary of Record Implementation

Constant time record access achieved by:
- representing properties by integers using \( idp \)
- mapping record property sets to integers using \( idr \)
- record access through arrays using lookup table \( p \)
Overview of Next Lecture

- Imperative Programming: The language imPL