WWU
MÜNSTER

ERCIS
European
Research
Center for
Information
Systems

An Operational Semantics for Constraint-logic Imperative Programming

Münster Logic-Imperative Programming Language

WFLP / Declare 2017

Jan C. Dageförde and Herbert Kuchen, University of Münster

September 22, 2017

Semantics of Muli

Münster Logic-Imperative Language (“Muli”)

Integration of constraint-logic programming and object-oriented programming

- Industry applications (occasionally) require search
- “Traditional” integrations tedious and error-prone
  (e.g. Java $\leftrightarrow$ Prolog: JNI, file-based)
- Muli: Integration on language level; specialised VM
- Language extends Java; symbolic JVM
- Runtime: Non-deterministic execution, backtracking, encapsulated search
- Non-determinism changes semantics!

$\Rightarrow$ Resolve ambiguities by defining an operational semantics

An Operational Semantics for Constraint-logic Imperative Programming

Concepts

- Free variables (and fields)

```
int y free;
```

- Constraints from evaluation of conditionals

```
while (y > 5) { [...] }
```

- Encapsulated search allowing non-determinism
- Search regions as lambda/method reference
- Solutions (including exceptions)

```
public static void log8() {
    int i = Muli.getOneSolution(() -> log_nd(8)); }

public static void powersOf2(String[] args) {
    Stream<Solution> powers = Muli.getAllSolutions(() -> {
        int x free; return pow(2, x); }); }
```

Demonstration

Example: Non-deterministic computation of $\log_2(x)$

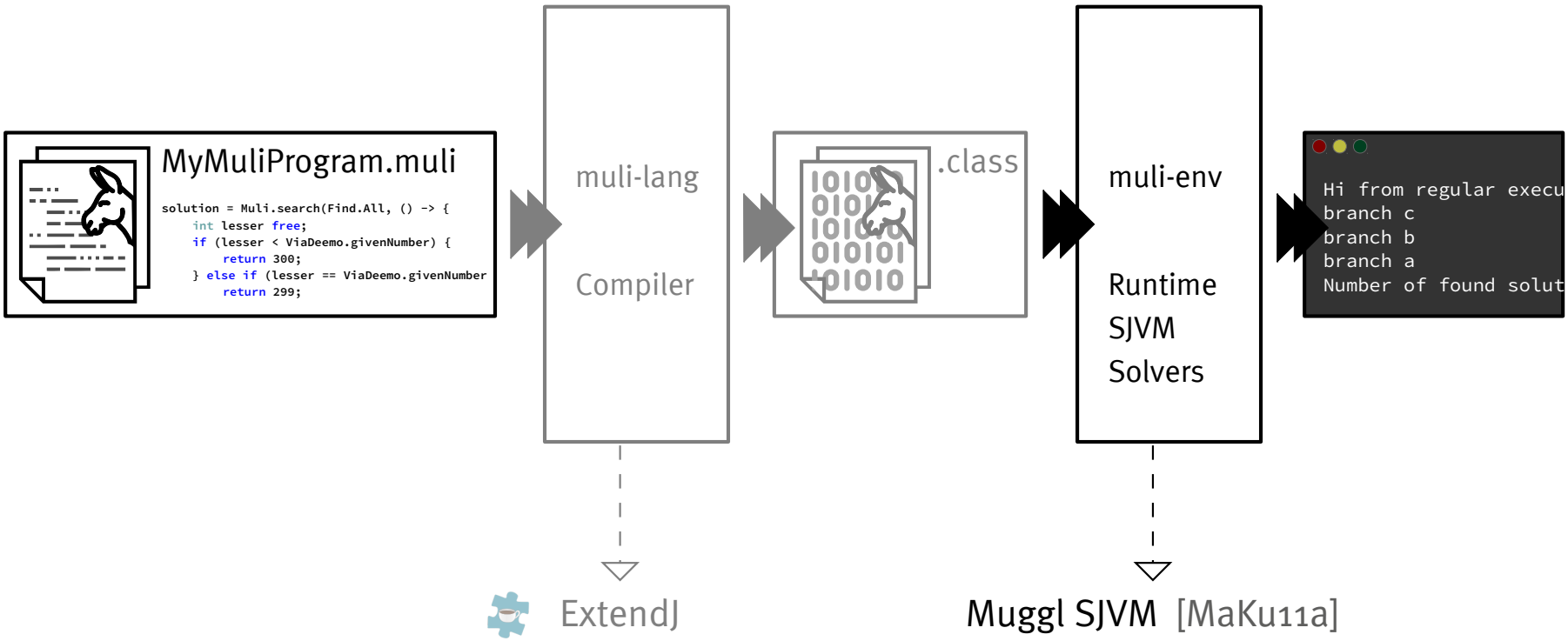
```
public static void main(String[] args) {
    int i = Muli.getOneSolution(() -> log_nd(8)); }
public static int log_nd(int x) {
    int y free;
    if (pow(2,y) == x) return y;
    else fail; }

public static int pow(int b, int y) {
    int i = 0; int r = 1;
    while (i < y) {
        r = r * b;
        i = i + 1; }
    return r; }
```

⇒ Minimal syntactic extension compared to Java

or (among others):

- (Dynamic) test case generation (incl. integration tests)
- Search problems from operations research
- Layout management

Muli


MyMuliProgram.muli
solution = Muli.search(Find.All, () -> {
int lesser free;
if (lesser < ViaDeemo.givenNumber) {
return 300;
} else if (lesser == ViaDeemo.givenNumber
return 299;
muli-lang
Compiler
.class
muli-env
Runtime
SJVM
Solvers
Hi from regular execu
branch c
branch b
branch a
Number of found solut
ExtendJ
Muggl SJVM [MaKu11a]

Focus: Imperative Subset of Muli

- Omit classes, OO

$\Rightarrow$ Describe interactions imperative $\leftrightarrow$ constraint-logic programming

Arithmetic expressions

$$e ::= c \mid x \mid e_1 \oplus e_2 \mid m(e_1, \ldots, e_k)$$

where $c \in \mathbb{Z}$, $x \in \mathit{Var}$, $e_1, \ldots, e_k \in \mathit{AExpr}$, $\oplus \in \mathit{AOp}$, $m \in \mathcal{M}$, $k \in \mathbb{N}$

Boolean expressions

$$b ::= e_1 \odot e_2 \mid b_1 \otimes b_2 \mid \texttt{true} \mid \texttt{false}$$

where $e_1, e_2 \in \mathit{AExpr}$, $b_1, b_2 \in \mathit{BExpr}$, $\odot \in \mathit{ROp}$, $\otimes \in \mathit{BOp}$

Statements

$$s ::= ; \mid \texttt{int}\, x; \mid \texttt{int}\, x\, \texttt{free}; \mid x = e; \mid e; \mid \{s\} \mid s_1\, s_2 \mid \texttt{if}\, (b)\, s_1\, \texttt{else}\, s_2 \mid \texttt{while}\, (b)\, s \mid \texttt{return}\, e; \mid \texttt{fail};$$

where $x \in \mathit{Var}$, $e \in \mathit{AExpr}$, $b \in \mathit{BExpr}$, $s, s_1, s_2 \in \mathit{Stat}$

Non-Deterministic Operational Semantics of (Imperative) Muli

Reduction semantics; computations depend on

- Environment $Env = (Var \to \mathcal{A}) \cup (\mathcal{M} \to Var^* \times Stat)$, $\rho \in Env$
- State $\Sigma = \mathcal{A} \to (\{\bot\} \cup Tree(\mathcal{A}, \mathbb{Z}))$, $\sigma \in \Sigma$
- Constraint store $CS = \{\texttt{true}\} \cup Tree(\mathcal{A}, \mathbb{Z})$, $\gamma \in CS$

$$Tree(\mathcal{A}, \mathbb{Z}) = \mathcal{A} \cup \mathbb{Z} \cup \{\oplus(t_1, t_2) \mid t_1, t_2 \in \mathbb{B} \cup Tree(\mathcal{A}, \mathbb{Z}), \oplus \in Op\}$$

Semantics of Expressions

$$\to \subset (Expr \times Env \times \Sigma \times CS) \times ((\mathbb{B} \cup Tree(\mathcal{A}, \mathbb{Z})) \times \Sigma \times CS)$$

7 rules

Semantics of Statements

$$\rightsquigarrow \subset (Stat \times Env \times \Sigma \times CS) \times (Env \times \Sigma \times CS)$$

13 rules

Reduction Semantics Rules

Arithmetic Expressions, Control Structures

AOp2

$$\frac{\langle e_1, \rho, \sigma, \gamma\rangle \to (v_1, \sigma_1, \gamma_1),\ \ \langle e_2, \rho, \sigma_1, \gamma_1\rangle \to (v_2, \sigma_2, \gamma_2),\ \ \{v_1, v_2\} \nsubseteq \mathbb{Z}}{\langle e_1 \oplus e_2, \rho, \sigma, \gamma\rangle \to (\oplus(v_1, v_2), \sigma_2, \gamma_2)}$$

Key

$\rho \in Env$

$\sigma \in \Sigma$

$\gamma \in CS$

$\mathbf{If_t}$

$$\frac{\langle b, \rho, \sigma, \gamma\rangle \to (v, \sigma_1, \gamma_1),\ \ \gamma_1 \not\models \neg v,\ \ \langle s_1, \rho, \sigma_1, \gamma_1 \wedge v\rangle \rightsquigarrow (\rho_1, \sigma_2, \gamma_2)}{\langle \texttt{if}\ (b)\ s_1\ \texttt{else}\ s_2, \rho, \sigma, \gamma\rangle \rightsquigarrow (\rho_1, \sigma_2, \gamma_2)}$$

If_f, Wh_t and Wh_f analogous

Reduction Semantics Rules

Assignments, Substitution, Labeling

Assign

$$\frac{\langle e, \rho, \sigma, \gamma\rangle \to (v, \sigma_1, \gamma_1)}{\langle x = e, \rho, \sigma, \gamma\rangle \rightsquigarrow (\rho[x/\alpha_1], \sigma_1[\alpha_1/v], \gamma_1)}$$

Key

$\rho \in Env$

$\sigma \in \Sigma$

$\gamma \in CS$

Subst

$$\frac{\gamma \models \sigma(\alpha) == v, \;\; \langle s, \rho, \sigma[\alpha/v], \gamma\rangle \;\rightsquigarrow (\rho_1, \sigma_1, \gamma_1)}{\langle s, \rho, \sigma, \gamma\rangle \;\rightsquigarrow (\rho_1, \sigma_1, \gamma_1)}$$

Label

$$\frac{\gamma \not\models \sigma(\alpha) \neq v, \;\; \langle s, \rho, \sigma[\alpha/v], \gamma \wedge (\sigma(\alpha) == v)\rangle \;\rightsquigarrow (\rho_1, \sigma_1, \gamma_1)}{\langle s, \rho, \sigma, \gamma\rangle \;\rightsquigarrow (\rho_1, \sigma_1, \gamma_1)}$$

Discussion

- Formalisation uncovered (and resolved) ambiguities
  w. r. t. Interpretation of symbolic variables and assignments, consistency checks
- Evaluation of control structures depends on CS performance, may evaluate (actually) infeasible branches
  Options:
  1. Explicitly label at every branch
  2. Label after solution was found
- Single assignment to preserve constraints
- Sharing by self-reference of free variables
- Backtracking is implicit; implementation not prescribed
  (e.g. iterative deepening DFS)

Related Work

Existing formal semantics of Java, unsuited for our extensions
Featherweight Java (focus: type system soundness) [IgPW01], Java specifications (natural language) [GJS+15;LYBB15]

Integrations OO $\rightarrow$ LP, unsuited for most software engineers in industry
Oz [VBD+03], Visual Prolog [Sc10], Prolog++ [M094], Concurrent Prolog [ShTa83]

Integrations LP $\rightarrow$ OO, not as seamless
Logic Java [MaKu11b], P@J/tuProlog [CiVi07;CiVi08]

Integration LP $\leftrightarrow$ FP, inspirational
Curry [AHH+02]

Conclusion

Operational reduction semantics of extended (imperative) Java

- Logic variables
- Encapsulated search
- Constraint solving

Formal basis for SJVM implementation

- GPL, `https://github.com/wwu-pi/muli-env`

Future work

- Extend semantics by more Java features, esp. classes, inheritance

European
Research
Center for
Information
Systems

Jan C. Dageförde — dagefoerde@uni-muenster.de

https://keybase.io/dagefoerde

THE IS RESEARCH NETWORK

www.ercis.org

References

[AHH+02] Albert, E., Hanus, M., Huch, F., Oliver, J., Vidal, G.: An operational semantics for declarative multi-paradigm languages. Electronic Notes in Theoretical Computer Science 70(6), 65–86 (2002)

[CiVi07] Cimadamore, M., Viroli, M.: A prolog-oriented extension of java programming based on generics and annotations. In: Amaral, V., et al. (eds.) Proceedings PPPJ. ACM ICPS, vol. 272, pp. 197–202. ACM (2007)

[CiVi08] Cimadamore, M., Viroli, M.: Integrating java and prolog through generic methods and type inference. In: Wainwright, R.L., Haddad, H. (eds.) Proceedings of the 2008 SAC. pp. 198–205. ACM (2008)

[MaKu11a] Majchrzak, Tim A.; Kuchen, Herbert: Muggl: The Muenster Generator of Glass-box Test Cases. In: ERCIS Working Papers, Nr. 10 (2011)

[MaKu11b] Majchrzak, T.A., Kuchen, H.: Logic Java: Combining Object-Oriented and Logic Programming. In: WFLP 2011. pp. 122–137 (2011)

[Mo94] Moss, C.: Prolog++ - the power of object-oriented and logic programming. International series in logic programming, Addison-Wesley (1994)

[Sc10] Scott, R.: A Guide to Artificial Intelligence with Visual Prolog. Outskirts Press (2010)

[ShTa83] Shapiro, E., Takeuchi, A.: Object oriented programming in Concurrent Prolog. New Generation Computing 1(1), 25–48 (1983)

[VBD+03] Van Roy, P., Brand, P., Duchier, D., Haridi, S., Schulte, C., Henz, M.: Logic programming in the context of multiparadigm programming: the Oz experience. Theory and Practice of Logic Programming 3(6), 717–763 (2003)

References

[BoRo15] Bogdanas, D., Rosu, G.: K-Java: A Complete Semantics of Java. POPL '15 pp. 1–12 (2015)

[GJS+15] Gosling, J., Joy, B., Steele, G., Bracha, G., Buckley, A.: The Java® Language Specification - Java SE 8 Edition (2015), https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[IgPW01] Igarashi, A., Pierce, B.C., Wadler, P.: Featherweight Java: A Minimal Core Calculus for Java and GJ. ACM Trans. Program. Lang. Syst.z 23(3), 396–450 (2001)

[LYBB15] Lindholm, T., Yellin, F., Bracha, G., Buckley, A.: The Java® Virtual Machine Specification – Java SE 8 Edition (2015), https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf