Algebraic Effects and Handlers in Natural Language Interpretation

Jiří Maršík and Maxime Amblard

LORIA, UMR 7503, Université de Lorraine, CNRS, Inria, Campus Scientifique, F-54506 Vandœuvre-lès-Nancy, France

July 17, 2014

Objectives

1. Detailed semantics for a large-scale grammar of a natural language

2. Capturing the interactions of non-local (i.e. non-compositional) semantic phenomena
   - anaphora
   - in-situ quantification
   - event arguments
   - presupposition
   - intensionalization
   - extraction
   - ...

3. Multiple semantic phenomena in a single treatment without overly complicated types and terms

In-situ quantification

Barker (2002)

Mary read every book.
$\forall x.\mathbf{book}(x) \to \mathbf{read}(\mathbf{Mary}, x)$

$$\llbracket s \rrbracket = o$$
$$\llbracket np \rrbracket = (\iota \to o) \to o$$

$$\llbracket \text{READ} \rrbracket : \llbracket np \rrbracket \to \llbracket np \rrbracket \to \llbracket s \rrbracket$$
$$\llbracket \text{READ} \rrbracket : ((\iota \to o) \to o) \to ((\iota \to o) \to o) \to o$$
$$\llbracket \text{READ} \rrbracket = \lambda so.s(\lambda x.o(\lambda y.\mathbf{read}(x, y)))$$

Anaphora

de Groote (2006)

Mary$_1$ read her$_1$ favorite book.

$\mathbf{read}(\mathbf{Mary}, \mathbf{favorite\text{-}book}(\mathbf{Mary}))$

$$
\begin{aligned}
[\![s]\!] &= \overline{o} \\
&= \gamma \to (\gamma \to o) \to o \\
[\![np]\!] &= (\iota \to \overline{o}) \to \overline{o} \\
&= (\iota \to \gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o
\end{aligned}
$$

$$
\begin{aligned}
[\![\text{READ}]\!] &: [\![np]\!] \to [\![np]\!] \to [\![s]\!] \\
[\![\text{READ}]\!] &: ((\iota \to \overline{o}) \to \overline{o}) \to ((\iota \to \overline{o}) \to \overline{o}) \to \overline{o} \\
[\![\text{READ}]\!] &: ((\iota \to \gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o) \\
&\quad \to ((\iota \to \gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o) \\
&\quad \to \gamma \to (\gamma \to o) \to o \\
[\![\text{READ}]\!] &= \lambda so.s(\lambda x.o(\lambda ye\phi.\mathbf{read}(x, y) \wedge \phi e))
\end{aligned}
$$

Motivation

- non-local phenomena + compositionality = generalizing meaning (often by abstracting over some new parameter)
- more non-local phenomena ⇒ more parameters ⇒ more complexity
- most research focuses on single phenomena

Effects in Interpretation

semantic generalizations $\approx$ monads

(Shan 2002)

Montague's PTQ $\approx$ evaluation order $+$ continuations

(Barker 2002)

non-local phenomena $\approx$ computational effects
$\Rightarrow$ elegant explanation of their interactions

(Kiselyov 2008; Shan 2005)

Effects and Handlers

Introduction

- Effectful operation: throws an exception containing the supplied argument and the current continuation
- Handlers: capture the exceptions to implement the operations
  - e.g. just by applying the continuation to some result
- Type-and-effect system: like Java's checked exceptions

$$\frac{(\mathbf{op} : A \to B) \in E \qquad \Gamma \vdash V : A \qquad \Gamma, x : B \vdash_E M : C}{\Gamma \vdash_E \mathbf{op}\ V\ (\lambda x.M) : C}$$

(Kammar, Lindley, and Oury 2013)

Effects and Handlers

Advantages

- Easier to combine multiple effects in a single semantics (Cartwright and Felleisen 1994) (Kiselyov, Sabry, and Swords 2013)

$$[\![C_E]\!] = C + \sum_{(\mathbf{op}:A\to B)\in E} A \times [\![C_E]\!]^B$$

$$\begin{array}{rcl} \text{"effects + handlers"} & : & \text{"delimited continuations"} \\ & = & \\ \text{"while"} & : & \text{"goto"} \end{array}$$

(Bauer and Pretnar 2012)

Translating Dynamic Logic

Effects and Handlers

- Effectful operations

$$\begin{aligned}
\textbf{get} &: 1 \to \gamma^{\{\textbf{get}\}} \\
\textbf{fresh} &: 1 \to \iota^{\{\textbf{fresh}\}} \\
\textbf{assert} &: o \to 1^{\{\textbf{assert}\}} \\
\textbf{scope_over} &: ((\iota \to o) \to o) \to \iota^{\{\textbf{scope_over}\}} \\
\textbf{move} &: 1 \to \iota^{\{\textbf{move}\}}
\end{aligned}$$

- Handlers

$$\begin{aligned}
\mathit{drs} &: \gamma \to (o^{\{\textbf{get};\textbf{fresh};\textbf{assert}|\rho\}} \Rightarrow o^{\rho}) \\
\mathit{tensed_clause} &: o^{\{\textbf{scope_over}|\rho\}} \Rightarrow o^{\rho} \\
\mathit{extract} &: \alpha^{\{\textbf{move}|\rho\}} \Rightarrow (\iota \to \alpha^{\rho})
\end{aligned}$$

Translating Dynamic Logic

Logical Connectives

$$\overline{\exists}\, P = \lambda e \phi . \exists x . P x (x :: e) \phi$$

$$\overline{\neg}\, A = \lambda e \phi . \neg (A e (\lambda e' . \top)) \wedge \phi e$$

$$A \mathbin{\overline{\wedge}} B = \lambda e \phi . A e (\lambda e' . B e' \phi)$$

Translating Dynamic Logic

Logical Connectives

$$\overline{\exists}\, P = \lambda e \phi . \exists x . P x (x :: e) \phi$$

$$\overline{\exists}\, P = P\ (\textbf{fresh}\ ())$$

$$\overline{\neg}\, A = \lambda e \phi . \neg (A e (\lambda e' . \top)) \wedge \phi e$$

$$\overline{\neg}\, A = \neg (\texttt{with}\ \mathit{drs}\ (\textbf{get}\ ())\ \texttt{handle}\ A)$$

$$A \mathbin{\overline{\wedge}} B = \lambda e \phi . A e (\lambda e' . B e' \phi)$$

$$A \mathbin{\overline{\wedge}} B = A \wedge B$$

Translating Dynamic Logic

Logical Connectives

$$\overline{\exists}\, P = \lambda e\phi.\exists x.Px(x :: e)\phi$$

$$\overline{\exists}\, P = P\ (\mathbf{fresh}\ ())$$

$$\overline{\neg}\, A = \lambda e\phi.\neg(Ae(\lambda e'.\top)) \wedge \phi e$$

$$\overline{\neg}\, A = \neg(\texttt{with}\ \mathit{drs}\ (\mathbf{get}\ ())\ \texttt{handle}\ A)$$

$$A \mathbin{\overline{\wedge}} B = \lambda e\phi.Ae(\lambda e'.Be'\phi)$$

$$A \mathbin{\overline{\wedge}} B = A \wedge B$$

$$A \mathbin{\overline{\Rightarrow}} B = \overline{\neg}(A \mathbin{\overline{\wedge}} \overline{\neg}\, B)$$

$$\overline{\forall}\, P = \overline{\neg}\, \overline{\exists} x. \overline{\neg}\, Px$$

Translating Dynamic Logic

Lexical Items

$$[\![\textsc{she}]\!] = \lambda ke\phi.k(sel_{she}(e))e\phi$$

$$[\![\textsc{something}]\!] = \lambda k.\overline{\exists}x.(k\ x) = \lambda ke\phi.\ \exists x.kx(x :: e)\phi$$

$$[\![\textsc{every}]\!] = \lambda nk.\overline{\forall}x.(n\ x) \overline{\Rightarrow} (k\ x)$$

$$[\![\textsc{read}]\!] = \lambda SO.S(\lambda s.O(\lambda oe\phi.\mathbf{read}(s, o) \land \phi e))$$

Translating Dynamic Logic

Lexical Items

$$[\![\textsc{she}]\!] = \lambda k e \phi . k(sel_{she}(e)) e \phi$$

$$[\![\textsc{she}]\!] = \lambda k . k(sel_{she}(\mathbf{get}\ ()))$$

$$[\![\textsc{something}]\!] = \lambda k . \overline{\exists} x . (k\ x) = \lambda k e \phi .\ \exists x . k x (x :: e) \phi$$

$$[\![\textsc{something}]\!] = \lambda k . \overline{\exists} x . (k\ x) = \lambda k . k(\mathbf{fresh}\ ())$$

$$[\![\textsc{every}]\!] = \lambda n k . \overline{\forall} x . (n\ x) \overline{\Rightarrow} (k\ x)$$

$$[\![\textsc{every}]\!] = \lambda n k . \overline{\forall} x . (n\ x) \overline{\Rightarrow} (k\ x)$$

$$[\![\textsc{read}]\!] = \lambda S O . S(\lambda s . O(\lambda o e \phi . \mathbf{read}(s, o) \wedge \phi e))$$

$$[\![\textsc{read}]\!] = \lambda S O . S(\lambda s . O(\lambda o . \mathbf{read}(s, o)))$$

Translating Dynamic Logic

Lexical Items

$$[\![\textsc{she}]\!] = \lambda k e \phi . k(sel_{she}(e)) e \phi$$

$$[\![\textsc{she}]\!] = \lambda k . k(sel_{she}(\textbf{get }()))$$

$$[\![\textsc{she}]\!] = \{sel_{she}(\textbf{get }())\}$$

$$[\![\textsc{something}]\!] = \lambda k . \overline{\exists} x . (k\ x) = \lambda k e \phi .\ \exists x . k x (x :: e) \phi$$

$$[\![\textsc{something}]\!] = \lambda k . \overline{\exists} x . (k\ x) = \lambda k . k(\textbf{fresh }())$$

$$[\![\textsc{something}]\!] = \{\textbf{fresh }()\}$$

$$[\![\textsc{every}]\!] = \lambda n k . \overline{\forall} x . (n\ x) \mathrel{\overline{\Rightarrow}} (k\ x)$$

$$[\![\textsc{every}]\!] = \lambda n k . \overline{\forall} x . (n\ x) \mathrel{\overline{\Rightarrow}} (k\ x)$$

$$[\![\textsc{every}]\!] = \lambda n . \{\textbf{scope\_over }(\lambda k . \overline{\forall} x . (n\ x) \mathrel{\overline{\Rightarrow}} (k\ x))\}$$

$$[\![\textsc{read}]\!] = \lambda S O . S(\lambda s . O(\lambda o e \phi . \textbf{read}(s, o) \wedge \phi e))$$

$$[\![\textsc{read}]\!] = \lambda S O . S(\lambda s . O(\lambda o . \textbf{read}(s, o)))$$

$$[\![\textsc{read}]\!] = \lambda s_t o_t . \{\texttt{with } \textit{tensed\_clause} \texttt{ handle } \textbf{read}(s_t!, o_t!)\}$$

Translating Dynamic Logic

Lexical Items (cont'd)

$$[\![\textsc{some}]\!] = \lambda nk.\,\overline{\exists}\,x.(n\ x)\,\overline{\wedge}\,(k\ x)$$

$$[\![\textsc{who}]\!] = \lambda rnx.nx\,\overline{\wedge}\,r(\lambda k.kx)$$

Translating Dynamic Logic

Lexical Items (cont'd)

$$[\![\textsc{some}]\!] = \lambda nk.\,\overline{\exists}\,x.(n\ x)\,\overline{\wedge}\,(k\ x)$$

$$\begin{aligned}[\![\textsc{some}]\!] &= \lambda nk.\,\overline{\exists}\,x.(n\ x) \wedge (k\ x)\\ &= \lambda nk.k(\ \texttt{let}\ x = \textbf{fresh}\ ()\ \texttt{in}\\ &\qquad\qquad \texttt{let}\ () = \textbf{assert}\ (n\ x)\ \texttt{in}\\ &\qquad\qquad x\ )\end{aligned}$$

$$[\![\textsc{who}]\!] = \lambda rnx.nx\,\overline{\wedge}\,r(\lambda k.kx)$$

$$[\![\textsc{who}]\!] = \lambda rnx.nx \wedge r(\lambda k.kx)$$

Translating Dynamic Logic

Lexical Items (cont'd)

$$[\![\textsc{some}]\!] = \lambda nk.\,\overline{\exists}x.(n\ x)\,\overline{\wedge}\,(k\ x)$$

$$[\![\textsc{some}]\!] = \lambda nk.\,\overline{\exists}x.(n\ x) \wedge (k\ x)$$

$$= \lambda nk.k(\ \texttt{let}\ x = \textbf{fresh}\ ()\ \texttt{in}$$
$$\texttt{let}\ () = \textbf{assert}\ (n\ x)\ \texttt{in}$$
$$x\ )$$

$$[\![\textsc{some}]\!] = \lambda n.\{\ \textbf{scope\_over}\ (\lambda k.\,\overline{\exists}x.(n\ x) \wedge (k\ x))\ \}$$

$$= \lambda n.\{\ \texttt{let}\ x = \textbf{fresh}\ ()\ \texttt{in}$$
$$\texttt{let}\ () = \textbf{assert}\ (n\ x)\ \texttt{in}$$
$$x\ \}$$

$$[\![\textsc{who}]\!] = \lambda rnx.nx\,\overline{\wedge}\,r(\lambda k.kx)$$

$$[\![\textsc{who}]\!] = \lambda rnx.nx \wedge r(\lambda k.kx)$$

$$[\![\textsc{who}]\!] = \lambda rnx.nx \wedge rx$$

Treating Extraction as an Effect

$$[\![\text{WHO}]\!] : [\![s]\!]^{\{\textbf{move}|\rho\}} \to [\![n]\!] \to [\![n]\!]^{\rho}$$

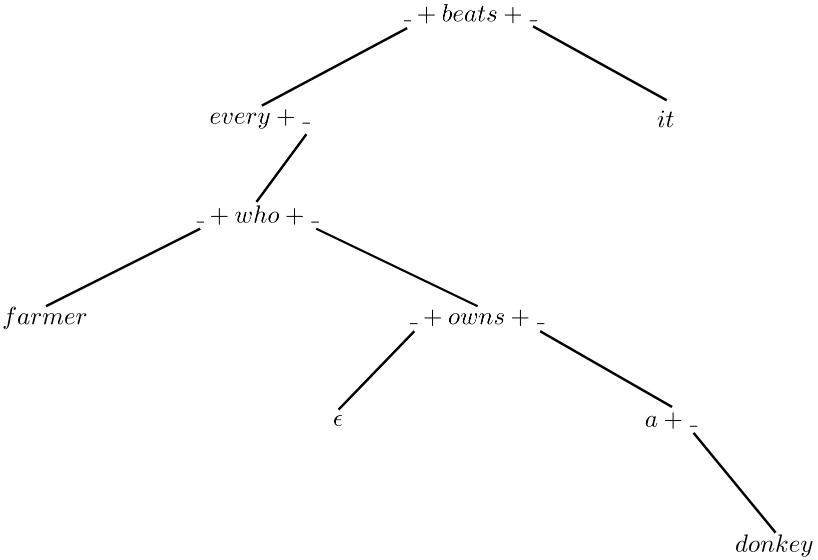
$$[\![\text{WHO}]\!] = \lambda r_t n.\lambda x.\ \texttt{let}\ r = \texttt{with}\ \mathit{extract}\ \texttt{handle}\ r_t!\ \texttt{in}$$
$$(n\ x) \wedge (r\ x)$$

$$r_t : o^{\{\textbf{move}|\rho\}}$$
$$r : \iota \to o^{\rho}$$
$$n : \iota \to o$$
$$x : \iota$$

$$[\![\epsilon]\!] : [\![np]\!]^{\{\textbf{move}\}}$$
$$[\![\epsilon]\!] = \{\textbf{move}\ ()\}$$

Example - Syntax

Every farmer who owns a donkey beats it.



Example - Semantics

Every farmer who owns a donkey beats it.

$\texttt{with}\ \mathit{tensed\_clause}\ \texttt{handle}\ \mathbf{beat}(\_, \_)$

$\mathbf{scope\_over}\ (\lambda k.\, \bar{\forall}\, x. \_(x) \mathbin{\bar{\to}} k(x))$

$\mathit{sel}_{it}(\mathbf{get}\ ())$

$\lambda x.\ \texttt{let}\ r = \texttt{with}\ \mathit{extract}\ \texttt{handle}\ \_\ \texttt{in}$

$\_(x) \wedge r(x)$

$\mathbf{farmer}$

$\texttt{with}\ \mathit{tensed\_clause}\ \texttt{handle}\ \mathbf{own}(\_, \_)$

$\mathbf{move}\ ()$

$\texttt{let}\ x = \mathbf{fresh}\ ()\ \texttt{in}$

$\texttt{let}\ () = \mathbf{assert}\ (\_(x))\ \texttt{in}$

x

$\mathbf{donkey}$

Conclusion

We have:

- motivated the use of algebraic effects and handlers in semantics.
- translated de Groote's continuation-based dynamic logic (de Groote 2006) to effects, reconstructing notions from DRT.
- treated extraction as an effect in interpretation instead of using hypothetical reasoning and lambda abstractions in the syntax.

Future Work

We would like to:

- show how effects and handlers apply to the other non-local phenomena (presupposition, event arguments, optional items, intensionalization).
- build a fragment that combines all of these.
- design a calculus with algebraic effects and handlers and a suitable evaluation order (CBV vs CBN).

Thank You

Questions?

Would you like to know more?

http://jirka.marsik.me/research/algebraic-effects-and-handlers-in-natural-language-interpretation

References

Barker, Chris (2002). "Continuations and the nature of quantification".

Bauer, Andrej and Matija Pretnar (2012). "Programming with algebraic effects and handlers".

Cartwright, Robert and Matthias Felleisen (1994). "Extensible denotational language specifications".

de Groote, Philippe (2006). "Towards a montagovian account of dynamics".

Kammar, Ohad, Sam Lindley, and Nicolas Oury (2013). "Handlers in action".

Kiselyov, Oleg (2008). "Call-by-name linguistic side effects".

Kiselyov, Oleg, Amr Sabry, and Cameron Swords (2013). "Extensible effects: an alternative to monad transformers".

Shan, Chung-chieh (2002). "Monads for natural language semantics".

Shan, Chung-chieh (2005). "Linguistic Side Effects".