Semantics of Advanced Data Types

Patricia Johann
Appalachian State University

June 17, 2021

Course Outline

Lecture 1: Syntax and semantics of ADTs and nested types ✓

Lecture 2: Syntax and semantics of GADTs ✓

Lecture 3: Parametricity for ADTs and nested types ✓

Lecture 4: Parametricity for GADTs

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Lecture 4:
Parametricity for GADTs

- In a parametric model, each type T[A] with a free type variable A has a set interpretation $T_0 : Set \to Set$ and a relational interpretation $T_1 : Rel \to Rel$ such that if $R : Rel(A, B)$ then $T_1 R : Rel(T_0 A, T_0 B)$ (and IEL holds).
- We argued last time that there are parametric models for calculi supporting ADTs and nested types.
- Can we also construct parametric models for calculi supporting GADTs?
- For this we need set and relational interpretations for GADTs as described above.
- Such interpretations exist for discrete semantics of GADTs.
- These satisfy the IEL (and AT).
- For fully functorial semantics of GADTs the situation is more complicated.

Set and Rel Interpretations of GADTs

- If

  data G : Set → Set where
     c : ∀{A : Set} → F A → G (K A)

  then

  - The set interpretation G_0 of G is μH_0, where $H_0\, J = Lan_{K_0}\, F_0$, i.e.,

  $$G_0 \;\cong\; \mu J.\, Lan_{K_0}\, F_0$$

  This left Kan extension is in Set, and K_0 interprets K and F_0 interprets F in Set.

  - The relational interpretation G_1 of G is μH_1, where $H_1\, J = Lan_{K_1}\, F_1$, i.e.,

  $$G_1 \;\cong\; \mu J.\, Lan_{K_1}\, F_1$$

  This left Kan extension is in Rel, and K_1 interprets K and F_1 interprets F in Rel.

Set and Rel Interpretations of GADTs

- If

  $$\begin{array}{l} \mathsf{data\ G\ \ :Set \to Set\ where} \\ \quad \mathsf{c : \forall \{A : Set\} \to F\,A \to G\,(K\,A)} \end{array}$$

  then

  - The set interpretation G_0 of G is μH_0, where $H_0\,J = Lan_{K_0}\,F_0$, i.e.,

  $$G_0 \;\cong\; \mu J.\, Lan_{K_0}\,F_0$$

  This left Kan extension is in Set, and K_0 interprets K and F_0 interprets F in Set.

  - The relational interpretation G_1 of G is μH_1, where $H_1\,J = Lan_{K_1}\,F_1$, i.e.,

  $$G_1 \;\cong\; \mu J.\, Lan_{K_1}\,F_1$$

  This left Kan extension is in Rel, and K_1 interprets K and F_1 interprets F in Rel.

Set and Rel Interpretations of GADTs

- If

$$\begin{array}{l} \mathsf{data\ G\ \ : Set \to Set\ where} \\ \quad \mathsf{c : \forall\{A : Set\} \to F\,A \to G\,(K\,A)} \end{array}$$

then

- The set interpretation G_0 of G is μH_0, where $H_0\,J = Lan_{K_0}\,F_0$, i.e.,

$$G_0 \;\cong\; \mu J.\,Lan_{K_0}\,F_0$$

This left Kan extension is in Set, and K_0 interprets K and F_0 interprets F in Set.

- The relational interpretation G_1 of G is μH_1, where $H_1\,J = Lan_{K_1}\,F_1$, i.e.,

$$G_1 \;\cong\; \mu J.\,Lan_{K_1}\,F_1$$

This left Kan extension is in Rel, and K_1 interprets K and F_1 interprets F in Rel.

A Problematic GADT

- Consider the GADT G whose single constructor c provides only an element for the instance $\mathsf{G}\,\top$

$$\begin{array}{l}\mathsf{data}\ \mathsf{G} : \mathsf{Set} \to \mathsf{Set}\ \mathsf{where}\\ \quad \mathsf{c} : \mathsf{G}\,\top\end{array}$$

- For this GADT, $\mathsf{F}\,\mathsf{U} = \top$ and $\mathsf{K}\,\mathsf{U} = \top$ for all $\mathsf{U} : \top$.

- The picture we should have in mind is:

$$\begin{tikzcd} Set^0 \arrow[rr, "F=\lambda u:Set^0.1"] \arrow[dr, "K=\lambda u:Set^0.1"'] & & Set \\ & Set \arrow[ur, "Lan_K\ F"'] & \end{tikzcd}$$

A Problematic GADT

- Consider the GADT G whose single constructor c provides only an element for the instance $G\,\top$

$$\begin{array}{l}\mathsf{data}\ G : Set \to Set\ \mathsf{where}\\ \quad c : G\,\top\end{array}$$

- For this GADT, $F\,U = \top$ and $K\,U = \top$ for all $U : \top$.

- The picture we should have in mind is:

$$\begin{tikzcd}
Set^0 \arrow[rr, "F=\lambda u:Set^0.1"] \arrow[dr, "K=\lambda u:Set^0.1"'] & & Set \\
& Set \arrow[ur, "Lan_K\ F"'] &
\end{tikzcd}$$

A Problematic GADT

- Consider the GADT G whose single constructor c provides only an element for the instance $\mathsf{G}\,\top$

$$
\begin{array}{l}
\mathsf{data\ G : Set \to Set\ where} \\
\quad \mathsf{c : G}\,\top
\end{array}
$$

- For this GADT, $\mathsf{F\,U} = \top$ and $\mathsf{K\,U} = \top$ for all $\mathsf{U} : \top$.

- The picture we should have in mind is:

$$
\begin{tikzcd}
Set^0 \arrow[rr, "F=\lambda u:Set^0.1"] \arrow[dr, "K=\lambda u:Set^0.1"'] & & Set \\
& Set \arrow[ur, "Lan_K\, F"'] &
\end{tikzcd}
$$

Computing the Set Interpretation for G

- The set interpretation G_0 of G is $G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A$.
- This is computed as

$$\Big(\bigcup_{U:Set^0,\; f:(\lambda u.1)\,U \to A} (\lambda u.1)\,U\Big) \,/ \sim \quad = \quad \Big(\bigcup_{U:Set^0,\; f:1\to A} 1\Big) \,/ \sim$$

- Here, U is the unique object of Set^0, $*$ is the unique element of the singleton set 1, and $\sim$ is the smallest equivalence relation such that $(U, f, *)$ and $(U, f', *)$ are related if

$$\begin{tikzcd}
(\lambda u.1)\,U \arrow[rr, "(\lambda u.1)\, id_U"] \arrow[dr, "f"'] & & (\lambda u.1)\,U \arrow[dl, "f'"] \\
& A &
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
1 \arrow[rr, "id_1"] \arrow[dr, "f"'] & & 1 \arrow[dl, "f'"] \\
& A &
\end{tikzcd}$$

commutes, i.e., if $f = f'$.
- Since the relation generating $\sim$ is already an equivalence relation,

$$(U, f, *) \sim (U, f', *) \text{ iff } f = f'$$

- So, for the fully functorial semantics,

$$G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A \;=\; \{f : 1 \to A\} = A$$

Computing the Set Interpretation for G

- The set interpretation G_0 of G is $G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A$.
- This is computed as

$$\Big(\bigcup_{U : Set^0,\; f : (\lambda u.1)\, U \to A} (\lambda u.1)\, U \Big) / \sim \quad = \quad \Big(\bigcup_{U : Set^0,\; f : 1 \to A} 1 \Big) / \sim$$

- Here, U is the unique object of Set^0, $*$ is the unique element of the singleton set 1, and $\sim$ is the smallest equivalence relation such that $(U, f, *)$ and $(U, f', *)$ are related if

$$\begin{tikzcd} (\lambda u.1)\, U \arrow[rr, "(\lambda u.1)\, id_U"] \arrow[dr, "f"'] & & (\lambda u.1)\, U \arrow[dl, "f'"] \\ & A & \end{tikzcd} \quad = \quad \begin{tikzcd} 1 \arrow[rr, "id_1"] \arrow[dr, "f"'] & & 1 \arrow[dl, "f'"] \\ & A & \end{tikzcd}$$

  commutes, i.e., if $f = f'$.
- Since the relation generating $\sim$ is already an equivalence relation,

$$(U, f, *) \sim (U, f', *) \text{ iff } f = f'$$

- So, for the fully functorial semantics,

$$G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A \;=\; \{f : 1 \to A\} = A$$

Computing the Set Interpretation for G

- The set interpretation G_0 of G is $G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A$.
- This is computed as

$$\Big(\bigcup_{U:Set^0,\; f:(\lambda u.1)\,U \to A} (\lambda u.1)\,U\Big) / \sim \quad = \quad \Big(\bigcup_{U:Set^0,\; f:1 \to A} 1\Big) / \sim$$

- Here, U is the unique object of Set^0, $*$ is the unique element of the singleton set 1, and $\sim$ is the smallest equivalence relation such that $(U, f, *)$ and $(U, f', *)$ are related if

$$\begin{tikzcd}
(\lambda u.1)\,U \arrow[rr, "(\lambda u.1)\, id_U"] \arrow[dr, "f"'] & & (\lambda u.1)\,U \arrow[dl, "f'"] \\
& A &
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
1 \arrow[rr, "id_1"] \arrow[dr, "f"'] & & 1 \arrow[dl, "f'"] \\
& A &
\end{tikzcd}$$

commutes, i.e., if $f = f'$.

- Since the relation generating $\sim$ is already an equivalence relation,

$$(U, f, *) \sim (U, f', *) \text{ iff } f = f'$$

- So, for the fully functorial semantics,

$$G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A \;=\; \{f : 1 \to A\} = A$$

Computing the Set Interpretation for G

- The set interpretation G_0 of G is $G_0 A \;=\; (Lan_{\lambda u.1}\,\lambda u.1)\,A$.
- This is computed as

$$\Big(\bigcup_{U:Set^0,\; f:(\lambda u.1)\,U\to A} (\lambda u.1)\,U\Big)/\sim \quad = \quad \Big(\bigcup_{U:Set^0,\; f:1\to A} 1\Big)/\sim$$

- Here, U is the unique object of Set^0, $*$ is the unique element of the singleton set 1, and $\sim$ is the smallest equivalence relation such that $(U, f, *)$ and $(U, f', *)$ are related if

$$\begin{tikzcd} (\lambda u.1)\,U \arrow[rr, "(\lambda u.1)\,id_U"] \arrow[dr, "f"'] & & (\lambda u.1)\,U \arrow[dl, "f'"] \\ & A & \end{tikzcd} \quad = \quad \begin{tikzcd} 1 \arrow[rr, "id_1"] \arrow[dr, "f"'] & & 1 \arrow[dl, "f'"] \\ & A & \end{tikzcd}$$

commutes, i.e., if $f = f'$.

- Since the relation generating $\sim$ is already an equivalence relation,

$$(U, f, *) \sim (U, f', *) \text{ iff } f = f'$$

- So, for the fully functorial semantics,

$$G_0 A \;=\; (Lan_{\lambda u.1}\,\lambda u.1)\,A \;=\; \{f : 1 \to A\} = A$$

Computing the Set Interpretation for G

- The set interpretation G_0 of G is $G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A$.
- This is computed as

$$\Big(\bigcup_{U:Set^0,\; f:(\lambda u.1)\,U \to A} (\lambda u.1)\,U\Big) \,/\, \sim \quad = \quad \Big(\bigcup_{U:Set^0,\; f:1 \to A} 1\Big) \,/\, \sim$$

- Here, U is the unique object of Set^0, $*$ is the unique element of the singleton set 1, and $\sim$ is the smallest equivalence relation such that $(U, f, *)$ and $(U, f', *)$ are related if

$$\begin{tikzcd}
(\lambda u.1)\,U \arrow[rr, "(\lambda u.1)\, id_U"] \arrow[dr, "f"'] & & (\lambda u.1)\,U \arrow[dl, "f'"] \\
& A &
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
1 \arrow[rr, "id_1"] \arrow[dr, "f"'] & & 1 \arrow[dl, "f'"] \\
& A &
\end{tikzcd}$$

  commutes, i.e., if $f = f'$.
- Since the relation generating $\sim$ is already an equivalence relation,

$$(U, f, *) \sim (U, f', *) \text{ iff } f = f'$$

- So, for the fully functorial semantics,

$$G_0 A \;=\; (Lan_{\lambda u.1}\, \lambda u.1)\, A \;=\; \{f : 1 \to A\} = A$$

Computing the Rel Interpretation for G

- The relational interpretation G_1 of G R is $G_1 R = (Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$.
- We need that if $R : Rel(A, B)$ then

$$(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R : Rel((Lan_{\lambda u.1}\, \lambda u.1)\, A, (Lan_{\lambda u.1}\, \lambda u.1)\, B) = Rel(A, B)$$

- Consider the relation $R = (1, 2, 1 \times 2)$, where 1×2 relates the single element $*$ of 1 to both elements of 2.
- For the IEL to hold for GADTs we expect the domain of $(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$ to be 1, but it is not!

Computing the Rel Interpretation for G

- The relational interpretation G_1 of G R is $G_1 R = (Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$.
- We need that if $R : Rel(A, B)$ then

$$(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R : Rel((Lan_{\lambda u.1}\, \lambda u.1)\, A, (Lan_{\lambda u.1}\, \lambda u.1)\, B) = Rel(A, B)$$

- Consider the relation $R = (1, 2, 1 \times 2)$, where 1×2 relates the single element $*$ of 1 to both elements of 2.
- For the IEL to hold for GADTs we expect the domain of $(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$ to be 1, but it is not!

Computing the Rel Interpretation for G

- The relational interpretation G_1 of G R is $G_1 R = (Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$.
- We need that if $R : Rel(A, B)$ then

$$(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R : Rel((Lan_{\lambda u.1}\, \lambda u.1)\, A, (Lan_{\lambda u.1}\, \lambda u.1)\, B) = Rel(A, B)$$

- Consider the relation $R = (1, 2, 1 \times 2)$, where 1×2 relates the single element $*$ of 1 to both elements of 2.
- For the IEL to hold for GADTs we expect the domain of $(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$ to be 1, but it is not!

Computing the Rel Interpretation for G

- The relational interpretation G_1 of G R is $G_1 R = (Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$.
- We need that if $R : Rel(A, B)$ then

$$(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R : Rel((Lan_{\lambda u.1}\, \lambda u.1)\, A, (Lan_{\lambda u.1}\, \lambda u.1)\, B) = Rel(A, B)$$

- Consider the relation $R = (1, 2, 1 \times 2)$, where 1×2 relates the single element $*$ of 1 to both elements of 2.
- For the IEL to hold for GADTs we expect the domain of $(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$ to be 1, but it is not!

IEL Does Not Hold for GADTs

- We can compute the domain of $(Lan_{\lambda u.Eq_1}\,\lambda u.Eq_1)\,R$ as

$$\begin{aligned}
\pi_1((Lan_{\lambda u.Eq_1}\,\lambda u.Eq_1)\,R) &= \pi_1(\varinjlim_{U:Rel^0,m:(\lambda u.Eq_1)\,U\to R}(\lambda u.Eq_1)\,R)\\
&= \pi_1(\varinjlim_{U:Rel^0,m:Eq_1\to R}\,Eq_1)\\
&= \varinjlim_{U:Rel^0,m:Eq_1\to R}\,(\pi_1\,Eq_1)\\
&= \varinjlim_{U:Rel^0,m:Eq_1\to R}\,1\\
&= \bigcup_{U:Rel^0,\,m:Eq_1\to R}\,1\,/\,\approx
\end{aligned}$$

- Here, U is the unique object of Rel^0, $*$ is the unique element of the singleton set 1, and $\approx$ is the smallest equivalence relation such that $(U,m,*)$ and $(U,m',*)$ are related if

$$\begin{tikzcd}
(\lambda u.Eq_1)\,U \arrow[rr, "(\lambda u.Eq_1)\,id_U"] \arrow[dr, "m"'] & & (\lambda u.Eq_1)\,U \arrow[dl, "m'"] \\
& R &
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
Eq_1 \arrow[rr, "id_{Eq_1}"] \arrow[dr, "m"'] & & Eq_1 \arrow[dl, "m'"] \\
& R &
\end{tikzcd}$$

commutes, so $(U,m,*) \approx (U,m',*)$ iff $m = m'$.

- So for the fully functorial semantics,

$$\pi_1(G_1 R) = (Lan_{\lambda u.Eq_1}\,\lambda u.1)\,R = \{m : Eq_1 \to R\} = \{(!,k_0),(!,k_1)\}$$

where k_0 and k_1 send $*$ to the two different elements of 2.

IEL Does Not Hold for GADTs

- We can compute the domain of $(Lan_{\lambda u.Eq_1}\ \lambda u.Eq_1)\ R$ as

$$
\begin{aligned}
\pi_1((Lan_{\lambda u.Eq_1}\ \lambda u.Eq_1)\ R) \quad &= \quad \pi_1(\varinjlim_{U:Rel^0,\, m:(\lambda u.Eq_1)\ U\to R}(\lambda u.Eq_1)\ R) \\
&= \quad \pi_1(\varinjlim_{U:Rel^0,\, m:Eq_1\to R}\ Eq_1) \\
&= \quad \varinjlim_{U:Rel^0,\, m:Eq_1\to R}\ (\pi_1\, Eq_1) \\
&= \quad \varinjlim_{U:Rel^0,\, m:Eq_1\to R}\ 1 \\
&= \quad \bigcup_{U:Rel^0,\ m:Eq_1\to R}\ 1\ /\ \approx
\end{aligned}
$$

- Here, U is the unique object of Rel^0, $*$ is the unique element of the singleton set 1, and $\approx$ is the smallest equivalence relation such that $(U, m, *)$ and $(U, m', *)$ are related if

$$
\begin{tikzcd}
(\lambda u.Eq_1)\,U \arrow[rr, "(\lambda u.Eq_1)\ id_U"] \arrow[dr, "m"'] & & (\lambda u.Eq_1)\,U \arrow[dl, "m'"] \\
& R &
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
Eq_1 \arrow[rr, "id_{Eq_1}"] \arrow[dr, "m"'] & & Eq_1 \arrow[dl, "m'"] \\
& R &
\end{tikzcd}
$$

  commutes, so $(U, m, *) \approx (U, m', *)$ iff $m = m'$.

- So for the fully functorial semantics,

$$\pi_1(G_1 R) \ = \ (Lan_{\lambda u.Eq_1}\ \lambda u.1)\ R \ = \ \{m : Eq_1 \to R\} \ = \ \{(!, k_0), (!, k_1)\}$$

  where k_0 and k_1 send $*$ to the two different elements of 2.

IEL Does Not Hold for GADTs

- We can compute the domain of $(Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R$ as

$$\begin{array}{rcl}
\pi_1((Lan_{\lambda u.Eq_1}\, \lambda u.Eq_1)\, R) & = & \pi_1(\varinjlim_{U:Rel^0, m:(\lambda u.Eq_1)\, U \to R} (\lambda u.Eq_1)\, R) \\
& = & \pi_1(\varinjlim_{U:Rel^0, m:Eq_1 \to R}\, Eq_1) \\
& = & \varinjlim_{U:Rel^0, m:Eq_1 \to R}\, (\pi_1\, Eq_1) \\
& = & \varinjlim_{U:Rel^0, m:Eq_1 \to R}\, 1 \\
& = & \bigcup_{U:Rel^0,\, m:Eq_1 \to R} 1 \;/\; \approx
\end{array}$$

- Here, U is the unique object of Rel^0, $*$ is the unique element of the singleton set 1, and $\approx$ is the smallest equivalence relation such that $(U, m, *)$ and $(U, m', *)$ are related if

$$\begin{tikzcd}
(\lambda u.Eq_1)\, U \arrow[rr, "(\lambda u.Eq_1)\, id_U"] \arrow[dr, "m"'] & & (\lambda u.Eq_1)\, U \arrow[dl, "m'"] \\
& R &
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
Eq_1 \arrow[rr, "id_{Eq_1}"] \arrow[dr, "m"'] & & Eq_1 \arrow[dl, "m'"] \\
& R &
\end{tikzcd}$$

commutes, so $(U, m, *) \approx (U, m', *)$ iff $m = m'$.

- So for the fully functorial semantics,

$$\pi_1(G_1 R) \;=\; (Lan_{\lambda u.Eq_1}\, \lambda u.1)\, R \;=\; \{m : Eq_1 \to R\} \;=\; \{(!, k_0), (!, k_1)\}$$

where k_0 and k_1 send $*$ to the two different elements of 2.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Fully Functorial GADTs Do Not Admit Traditional Parametric Models

- Since the set and relational interpretations of GADTs are not appropriately fibred, the IEL cannot possibly hold.
- We conclude that fully functorial GADTs do not admit parametric models if GADTs are given a traditional relational semantics!
- We get the naturality consequences of parametricity functorial GADTs just from the functorial semantics.
- But if we can prove inhabitation results, or prove representation independence, or derive short cut fusion or deep induction rules for them, then it has to be from something other than parametricity.
- By contrast, discrete (syntax-only) GADTs have parametric models (but not non-trivial functoriality).
- So we can prove inhabitation results and prove representation independence and derive short cut fusion and deep induction rules for them. But we cannot derive (non-trivial) naturality consequences of parametricity for them.
- So there's a trade-off, and the choice we make has consequences.

Course Outline

Lecture 1: Syntax and semantics of ADTs and nested types ✓

Lecture 2: Syntax and semantics of GADTs ✓

Lecture 3: Parametricity for ADTs and (truly) nested types ✓

Lecture 4: Parametricity for GADTs ✓