8 Calcoli tipati e sistemi di tipi

In questa sezione introdurremo vari calcoli tipati, essi possono essere dati in due versioni: usando termini senza informazione di tipo, oppure usando termini con informazione di tipo (vi e' un certo grado di discrezionalita' quanta informazione di tipo esplicitare). Prima di considerare calcoli specifici, introduciamo un po' di terminologia e notazione per un generico linguaggio tipato L.

- $\tau \in T(L)$ espressioni di tipo
- $M \in \Lambda_T(L)$ termini con informazioni di tipo, $M \in \Lambda(L)$ termini senza informazione di tipo

  in generale un indice T indichera' la versione di L che usa termini con informazioni di tipo
- $|\_|: \Lambda_T(L) \to \Lambda(L)$ funzione di cancellazione dell'informazione di tipo
- $\Gamma \vdash M\colon \tau$ typing judgement, dove $\Gamma\colon Var \stackrel{fin}{\to} T(L)$ e' un typing context e $\mathrm{FV}(M) \subseteq dom(\Gamma)$
- $V \in \mathsf{Val}(L) \subseteq \Lambda(L)$ valori aperti; $V \in \mathsf{Val}_0(L) = \mathsf{Val}(L) \cap \Lambda_0(L)$ valori chiusi
- $M \Downarrow_e R$ semantica operazionale CBV con err, dove $M \in \Lambda_0(L)$ e $R \in \mathsf{Val}_0(L) + \{err\}$.

  La semantica operazionale con err serve a descrivere in modo preciso la classe di errori che si vorrebbero identificare a tempo di compilazione (mediante il type-checking). In certi casi puo' essere conveniente usare una semantica operazionale con ambiente: $\rho \vdash M \Downarrow_e R$, dove $\rho\colon Var \stackrel{fin}{\to} \mathsf{Val}_0(L)$ ambiente e $\mathrm{FV}(M) \subseteq dom(\rho)$.
- Dato un sistema (di type-checking) per derivare typing judgements, diciamo che vale la **correttezza** del type-checking relativamente alla semantica operazionale con err: "se M ha un tipo, allora non genera err a tempo di esecuzione", cioe' $\emptyset \vdash M\colon \tau$ e $M \Downarrow R$ implicano $R \not\equiv err$.

Descriveremo anche la semantica semplice dei tipi, nella quale un tipo τ e' interpretato da un sottoinsieme $[\![\tau]\!] \subseteq \mathsf{Val}_0$ di valori. Tale semantica permette di definire la soddisfacibilita' di un typing judgement $\Gamma \models M\colon \tau$ in funzione della sola semantica operazione, ed analogamente permette una interpretazione ovvia della relazione di sub-typing $\tau_1 \subseteq \tau_2$, cioe' $[\![\tau_1]\!] \subseteq [\![\tau_2]\!]$, e quindi validare la soundness di regole di inferenza per il sub-typing.

8.1 I tipi semplici

I tipi semplici sono quelli che tipicamente si incontrano in linguaggio di programmazione come il Pascal o Algol, cioe' con parametri di tipo procedure ma senza *polimorfismo*.

- $\tau \in T ::= \overline{\tau}\ C \mid \tau_1 \to \tau_2$, dove C e' un costruttore di tipo, p.e. $bool$ o $\tau\ list$.
- $M \in \Lambda ::= x \mid c \mid \lambda x.M \mid M_1 M_2$, dove c sono delle costanti (higher-order), p.e. if, nil, $cons$, fix.

  $M \in \Lambda_T ::= x \mid c_{\overline{\tau}} \mid \lambda x\colon \tau.M \mid M_1 M_2$ e' l'insieme dei termini con informazioni di tipo, e la funzione $|\_|: \Lambda_T \to \Lambda$ di cancellazione della informazione di tipo e' data da: $|x| = x$, $|c_{\overline{\tau}}| = c$, $|\lambda x\colon \tau.M| = \lambda x.|M|$, $|M_1 M_2| = |M_1||M_2|$.
- Le regole di inferenza per derivare typing judgements con informazioni di tipo nei termini sono

  $$(\mathrm{x})\ \Gamma \vdash_T x\colon \Gamma(x) \qquad (\mathrm{c})\ \Gamma \vdash_T c\colon \tau_c \qquad (\to\mathrm{I})\ \frac{\Gamma, x\colon \tau_1 \vdash_T M\colon \tau_2}{\Gamma \vdash_T \lambda x\colon \tau_1.M\colon \tau_1 \to \tau_2} \qquad (\to\mathrm{E})\ \frac{\begin{array}{c}\Gamma \vdash_T M_1\colon \tau_1 \to \tau_2 \\ \Gamma \vdash_T M_2\colon \tau_1\end{array}}{\Gamma \vdash_T M_1 M_2\colon \tau_2}$$

  alcuni esempi di tipi per le costanti sono:

  - $T, F\colon bool$ e $if_\tau\colon bool \to \tau \to \tau \to \tau$
  - $nil_\tau\colon \tau\ list$, $cons_\tau\colon \tau \to \tau\ list \to \tau\ list$ e $case_{\tau,\sigma}\colon \tau\ list \to \sigma \to (\tau \to \tau\ list \to \sigma) \to \sigma$, dove $case$ corrisponde al pattern matching per il tipo $\tau\ list$
  - $fix_\tau\colon (\tau \to \tau) \to \tau$, che serve per modellare le definizioni ricorsive.

  La seguente regola di **sostitutivita'** risulta essere *ammissibile*, cioe' se la premessa e' derivabile anche la conclusione lo e', e quindi puo' essere aggiunta senza modificare l'insieme dei typing judgements derivabili

  $$(\mathrm{sub})\ \frac{\Gamma \vdash M\colon \tau \quad \Gamma, x\colon \tau \vdash M'\colon \tau'}{\Gamma \vdash [M/x]M'\colon \tau'}$$

  Le regole di inferenza per derivare typing judgements senza informazioni di tipo nei termini si ottengono dalle precedenti applicando la funzione di cancellazione $|\_|$ ai termini M che compaiono nella regola.

- La semantica operazionale CBV con *err* $M \Downarrow R$ estende quella CBV precedentemente introdotta con regole aggiuntive per trattare le situazioni di errore. La daremo per i termini senza informazione di tipo, poiche' la semantica operazione non fa' alcun uso di tale informazione. Per semplicita' consideriamo il linguaggio con le sole costanti per il tipo *bool*, quindi l'insieme dei valori e' dato da $V \in \mathsf{Val}: := x \mid T \mid F \mid \lambda x.M$.

$$\text{(V)}\ V \Downarrow V \qquad \text{(app)}\ \frac{M_1 \Downarrow \lambda x.M \quad M_2 \Downarrow V \quad [V/x]M \Downarrow R}{M_1 M_2 \Downarrow R} \qquad \text{(if.T)}\ \frac{M \Downarrow T \quad M_1 \Downarrow R}{if(M, M_1, M_2) \Downarrow R}$$

$$\text{(app1)}\ \frac{M_1 \Downarrow \lambda x.M \quad M_2 \Downarrow err}{M_1 M_2 \Downarrow err} \qquad \text{(app2)}\ \frac{M_1 \Downarrow R \not\equiv \lambda x.M}{M_1 M_2 \Downarrow err} \qquad \text{(if1)}\ \frac{M \Downarrow R \not\equiv T, F}{if(M, M_1, M_2) \Downarrow err}$$

- La semantica di una espressione di tipo τ e' data da un sottoinsieme $[\![\tau]\!] \subseteq \mathsf{Val}_0$ definito per induzione su τ. La definizione di $[\![\tau]\!]$ fa' uso della semantica operazione con *err*, ma e' indipendente dai typing judgments.

  Dato $X \subseteq \mathsf{Val}_0$ definiamo $\widetilde{X} \stackrel{\Delta}{=} \{M \in \Lambda_0 | \forall R.M \Downarrow R \supset R \in X\}$. Intuitivamente $\widetilde{X}$ per l'insieme dei programmi $M \in \Lambda_0$ che in caso di terminazione producono un valore in X (e quindi non possono terminare con *err*).

  Data una semantica delle espressioni di tipo, diciamo che $\Gamma \models M\colon \tau$ vale $\stackrel{\Delta}{\Longleftrightarrow} \forall \rho \in [\![\Gamma]\!].([\rho]M) \in \widetilde{[\![\tau]\!]}$, dove $[\![\Gamma]\!] \stackrel{\Delta}{=} \{\rho\colon Var \stackrel{fin}{\to} \mathsf{Val}_0 | \forall x \in \mathrm{DV}(\Gamma).\rho(x) \in [\![\Gamma(x)]\!]\}$ e $[\rho]M \in \Lambda_0$ e' l'istanza di M ottenuta sostituendo ogni $x \in \mathrm{FV}(M)$ con $\rho(x) \in \mathsf{Val}_0$.

  Diamo il corrispettivo semantico di *bool* e dei costruttori *list* e $\to$, che quindi agiscono su insiemi $X, Y \subseteq \mathsf{Val}_0$

  - $bool \stackrel{\Delta}{=} \{T, F\}$
  - $X \to Y \stackrel{\Delta}{=} \{\lambda x.M \in \mathsf{Val}_0 | \forall V \in X.[V/x]M \in \widetilde{Y}\}$
  - $X\ list \stackrel{\Delta}{=} \{[V_i|i\colon n] | \forall i\colon n.V_i \in X\}$, dove $[V_i|i\colon n]$ e' una abbreviazione per $cons(V_0, \ldots, cons(V_{n-1}, nil)\ldots)$

  La semantica dei tipi, oltre a spiegarne il significato, e' anche utile per stabilire la **soundness** di regole per derivare typing judgements (nello stesso modo in cui l'interpretazione della logica del primo ordine, permette di validare assioni e regole di inferenza). Una volta stabilita la soundness, cioe' $\Gamma \vdash M\colon \tau$ derivabile implica $\Gamma \models M\colon \tau$ vera, la correttezza discende come un facile corollario. Infatti, dalla definizione di $\Gamma \models M\colon \tau$ e' immediato dimostrare che $\emptyset \models M\colon \tau$ e $M \Downarrow R$ implicano $R \not\equiv err$.

Dimostriamo la correttezza delle regole di type-checking passando attraverso una dimostrazione di soundness. In seguito illustreremo una tecnica piu' diretta per dimostrare la correttezza (ma anche meno informativa).
Per semplicita' ci limiteremo a dimostrare la soundness delle regole ($\to$I) e ($\to$E).

- Per ($\to$I) dobbiamo dimostrare che $\Gamma, x\colon \tau_1 \models M\colon \tau_2$ implica $\Gamma \models \lambda x\colon \tau_1.M\colon \tau_1 \to \tau_2$. Per semplicita' supponiamo che Γ sia vuoto.

  Dalla definizione di $x\colon \tau_1 \models M\colon \tau_2$ abbiamo che $\forall V\colon [\![\tau_1]\!].[V/x]M \in \widetilde{[\![\tau_2]\!]}$, ma questa e' esattamente la definizione di $(\lambda x.M) \in [\![\tau_1]\!] \to [\![\tau_2]\!]$, che e' banalmente equivalente alla validita di $\lambda x\colon \tau_1.M\colon \tau_1 \to \tau_2 \models$.

- Per ($\to$E) dobbiamo dimostrare che $\Gamma \models M_1\colon \tau_1 \to \tau_2$ e $\Gamma \models M_2\colon \tau_1$ implicano $\Gamma \models M_1 M_2\colon \tau_2$. Anche in questo caso supponiamo che Γ sia vuoto.

  Espandendo la definizione di $\models M_1 M_2\colon \tau_2$, quello che dobbiamo dimostrare e': $M_1 M_2 \Downarrow R$ implica $R \in [\![\tau_2]\!]$. Facendo una analisi delle possibili regole per derivare $M_1 M_2 \Downarrow R$ si vede che l'unica regola possibile e' (app). Infatti se fosse (app2) si avrebbe $M_1 \Downarrow R_1 \not\equiv \lambda x.M$ che contraddice $\Gamma \models M_1\colon \tau_1 \to \tau_2$, e se fosse (app1) si avrebbe $M_2 \Downarrow err$ che contraddice $\Gamma \models M_2\colon \tau_1$. Quindi abbiamo $M_1 \Downarrow \lambda x.M$, $M_2 \Downarrow V$ e $[V/x]M \Downarrow R$. Ma per le assunzioni si ha che $\lambda x.M \in [\![\tau_1]\!] \to [\![\tau_2]\!]$ e $V \in [\![\tau_2]\!]$, e percio' $[V/x]M \in \widetilde{[\![\tau_2]\!]}$ e infine $R \in [\![\tau_2]\!]$.

8.2 I tipi semplici con variabili

Una estensione piccola, ma significativa, dei tipi semplici consiste nell'aggiunta delle variabili di tipo. Questa estensione fornisce un modo di esprimere il polimorfismo di un termine M assegnandogli un tipo τ con variabili di tipo. Sostituendo le variabili di tipo in τ con espressioni di tipo si ottengono altri tipi (piu' specifici) per M. Questa estensione ci permette di illustrare l'algoritmo W di type-inference dovuto ad Hindley-Milner in forma semplificata.

- $\tau \in T: := \overline{\tau}\ C \mid \tau_1 \to \tau_2 \mid X$, dove C e' un costruttore di tipo (come illustrato in precedenza)

- Tutte le altre definizioni ricalcano nel modo ovvio quelle gia' date per i tipi semplici. In particolare, la definizioni di Λ e la semantica operazionale CBV $M \Downarrow R$ per i termini senza informazioni di tipo sono invariate, mentre i typing judgements $\Gamma \vdash M : \tau$ e le regole di type-checking sono uguali modulo l'estensione di T.

  La seguente regola di **sostitutivita'** per variabili di tipo risulta essere ammissibile

  $$(\mathrm{Sub})\ \frac{\Gamma \vdash M : \tau}{[S]\Gamma \vdash M : [S]\tau}$$

  dove $S : TVar \stackrel{fin}{\to} T$ e' una sostituzione di variabili di tipo con espressioni di tipo, $[S]\tau$ e' l'istanza di τ ottenuta applicando la sostituzione S (si conviene che $S[X] = X$ se $X \notin dom(S)$) e $[S]\Gamma$ e' il typing context Γ' con lo stesso dominio di Γ t.c. $\Gamma'(x) = [S](\Gamma(x))$.

- Adesso la semantica $[\![\tau]\!]_\eta$ di una espressione di tipo τ dipendera' anche da un ambiente $\eta : TVar \stackrel{fin}{\to} \mathcal{P}(\mathsf{Val}_0)$ che fornisce l'interpretazione di tutte le variabili di tipo in $\mathrm{FV}(\tau)$, cioe' $\mathrm{FV}(\tau) \subseteq dom(\eta)$. Modulo cio' la definizione e' uguale al caso semplice, ovvero $[\![\tau_1 \to \tau_2]\!]_\eta = [\![\tau_1]\!]_\eta \to [\![\tau_2]\!]_\eta$, etc.

  Corrispondentemente la definizione di "$\Gamma \models M : \tau$ vale" richiede una ulteriore quantificazione sugli ambienti η, cioe' $\forall \rho \in [\![\Gamma]\!]_\eta.([\rho]M) \in \widetilde{[\![\tau]\!]}_\eta$ per ogni $\eta : TVar \stackrel{fin}{\to} \mathcal{P}(\mathsf{Val}_0)$ t.c. $\mathrm{FV}(\Gamma, \tau) \subseteq dom(\eta)$.

L'algoritmo W di Hindley-Milner puo' essere descritto nella seguente maniera

- input: $\Gamma : Var \stackrel{fin}{\to} T$ e $M \in \Lambda$ t.c. $\mathrm{FV}(M) \subseteq dom(\Gamma)$
- output: fail oppure una coppia (S, τ) dove $S : \mathrm{FV}(\Gamma) \to T$ e $\tau \in T$
- $W(\Gamma; M)$ fallisce sse non esistono S e τ t.c. $[S]\Gamma \vdash M : \tau$ sia derivabile
- altrimenti $W(\Gamma; M)$ restituisce una coppia (S, τ) t.c. $[S]\Gamma \vdash M : \tau$ e' derivabile, ed inoltre se $S' : \mathrm{FV}(\Gamma) \to T$ e $[S']\Gamma \vdash M : \tau'$ e' derivabile, allora esiste S'' t.c. $S' = [S'']S$ e $\tau' = [S'']\tau$. Dove $[S_2]S_1$ e' la sostituzione S_3 con lo stesso dominio di S_1 t.c. $S_3(X) = [S_2](S_1(X))$.

Si osservi che per inferire il **tipo piu' generale** τ di M basta prendere $\Gamma : \mathrm{FV}(M) \hookrightarrow TVar$. L'algortimo Wprocede per induzione sulla struttura di M.

- $W(\Gamma; x) = (Id_{\mathrm{FV}(\Gamma)}, \Gamma(x))$, dove Id_U indica la sostituzione identica con dominio $U \subseteq_{fin} TVar$.
- $W(\Gamma; \lambda x.M) = (S|_{\mathrm{FV}(\Gamma)}, S(X) \to \tau)$ se $(S, \tau) = W(\Gamma, x : X; M)$, dove $X \in TVar$ e' *new*, $S|_U$ e' la restrizione di S al dominio U, ed inoltre si suppone che $x \notin dom(\Gamma)$ (se necessario si puo' effettuare una ridenominazione delle variabili legate). Per dare una definizione precisa di "X new" occorrerebbe passare all'algoritmo W un terzo parametro $U \subseteq_{fin} TVar$, per indicare le variabili sinora usate.
- $W(\Gamma; M_1 M_2) = ([S_3][S_2]S_1, S_3(X))$ se $(S_1, \tau_1) = W(\Gamma; M_1)$, $(S_2, \tau_2) = W([S_1]\Gamma; M_2)$ e $S_3 = mgu([S_2]\tau_1, \tau_2 \to X)$, dove $X \in TVar$ e' *new* e $mgu(\tau, \tau')$ e' il **most general unifier** delle espressioni di tipi τ e τ'. La non esistenza di $mgu(\tau, \tau')$ e' il motivo per cui l'algoritmo W puo' fallire.
- $W(\Gamma; c) = (Id_{\mathrm{FV}(\Gamma)}, \tau_c)$, dove τ_c e' un **tipo piu' generale** per la costante c che usa solo varibili di tipo new, p.e. $\tau_{nil} = X\ list$ con X new. Questo e' uno dei casi in cui la coppia (S, τ) restituita da W non soddisfa la proprieta' $\mathrm{FV}(\tau) \subseteq \mathrm{FV}(S)$.

Consideriamo alcuni esempi di tipi inferiti da W per termini chiusi.

- $W(\emptyset; \lambda x.x) = (\emptyset, X \to X)$ con X new
- $W(\emptyset; (\lambda x.x)(\lambda x.x)) = (\emptyset, Y \to Y)$ con Y new, le due occorrenze di $(\lambda x.x)$ ricevono tipi diversi $X \to X$ e $Y \to Y$, ed infine si deve calcolare $S_3 = mgu(X \to X, (Y \to Y) \to Z)$, quindi il tipo calcolato e' $S_3(Z) = Y \to Y$.
- $W(\emptyset; \lambda x.xx)$ fallisce, infatti $W(x : X, x) = (Id, X)$ ma $mgu(X, X \to Z)$ fallisce.
- Come esercizio far vedere che (modulo i nomi delle variabili) W inferisce i seguenti tipi
  - $\lambda g, f, x.g(fx) : (Y \to Z) \to (X \to Y) \to X \to Z$
  - $fix(\lambda x.x) : X$
  - $cons(nil, nil) : X\ list\ list$
  - $map = fix(\lambda m, f, l.case_{list}(l, nil, \lambda x, l.cons(fx, l))) : (x \to Y) \to X\ list \to Y\ list$

8.3 Tipi polimorfi

L'aggiunta dei tipi polimorfi permette di tipare molti piu' termini $M \in \Lambda$. Tuttavia per tale estensione non esiste un analogo dell'algoritmo W, anzi e' stato dimostrato che e' indecidibile se $\Gamma \vdash M\colon \tau$ e' derivabile (ovviamente quando si usano i termini con informazioni di tipo la derivabilita' di $\Gamma \vdash_T M\colon \tau$ e' facilmente decidibile).

- $\tau \in T ::= \overline{\tau}\ C \mid \tau_1 \to \tau_2 \mid X \mid \forall X.\tau$, dove C e' un costruttore di tipo (come illustrato in precedenza)
- $M \in \Lambda_T ::= x \mid c \mid \lambda x\colon \tau.M \mid M_1 M_2 \mid \Lambda X.M \mid M[\tau]$, si osservi che le costanti polimorfe non hanno piu' bisogno di essere viste come famiglie di costanti indiciate da tipi. La definizione di $M \in \Lambda$ e' invariata, mentre la funzione di cancellazione dell'informazione di tipo e' estesa nel modo seguente: $|\Lambda X.M| = |M[\tau]| = |M|$.
- Vi sono due nuove regole di inferenza per derivare typing judgements (con informazioni di tipo nei termini)

  $$(\forall\mathrm{I})\ \frac{\Gamma \vdash_T M\colon \tau}{\Gamma \vdash_T \Lambda X.M\colon \forall X.\tau}\ X \notin \mathrm{FV}(\Gamma) \qquad (\forall\mathrm{E})\ \frac{\Gamma \vdash_T M\colon \forall X.\tau}{\Gamma \vdash_T M[\tau']\colon [\tau'/X]\tau}\ \tau' \in T$$

  inoltre vi e' un modo piu' naturale di tipare le costanti polimorfe:

  - $T, F\colon bool$ e $if\colon \forall X.bool \to X \to X \to X$
  - $nil\colon \forall X.X\ list$, $cons\colon \forall X.X \to X\ list \to X\ list$ e $case\colon \forall X,Y.X\ list \to Y \to (X \to X\ list \to Y) \to Y$
  - $fix\colon \forall X.(X \to X) \to X$

  Usando le nuove regole alcuni termini che prima non erano tipabili adesso lo divetano, p.e. $\lambda x.xx$ ha sia tipo $(\forall X.X \to X) \to X \to X$ che tipo $(\forall X.X) \to Y$ come si vede piu' chiaramente aggiungendo informazione di tipo

  $$\lambda x\colon (\forall X.X \to X).x[X \to X]\ (x[X]) \qquad \lambda x\colon (\forall X.X).x[X \to Y]\ (x[X])$$

  $\overline{X}\colon T \vdash \tau\colon T$, dove $\mathrm{FV}(\tau) \subseteq \overline{X}$

  $\overline{X}\colon T; \Gamma \vdash M\colon \tau$, dove $\mathrm{FV}(\Gamma,\tau) \subseteq \overline{X}$ e $\mathrm{FV}(M) \subseteq dom(\Gamma)$ (per termini senza tipi)
- La semantica operazionale CBV con $err\ M \Downarrow R$ e' invariata.
- La semantica $[\![\tau]\!]_\eta$ estende quella data per i tipi semplici con variabili di tipo per trattare il caso $\forall X.\tau$ dei tipi polimorfi $[\![\forall X.\tau]\!]_\eta = \cap\{[\![\tau]\!]_{\eta[X:=A]} | A \subseteq \mathsf{Val}_0\}$. Alternativamente possiamo dare la semantica dei tipi polimorfi usando il corrispettivo semantico di $\forall$

  - $\forall(F) = \cap\{F(A) | A \subseteq V_0\}$, dove $F\colon \mathcal{P}(\mathsf{Val}_0) \to \mathcal{P}(\mathsf{Val}_0)$.

  La definizione di "$\Gamma \models M\colon \tau$ vale" e' uguale a quella data nel caso di tipi semplici con variabili di tipo, e la dimostrazione di **soundness** si estende facilmente. Si osservi che in presenza di tipi polimorfi si hanno relazioni di sub-typing non banali, infatti $(\forall X.\tau) \subseteq [\tau'/X]\tau$ per qualunque τ e τ'.

8.4 I tipi di ML

Il sistema di tipi di ML aggiunge una forma ristretta di tipi polimorfi (detti schemi di tipo) senza pero' perdere le proprieta' dell'algoritmo W di type-inference.

- $\tau \in T ::= \overline{\tau}\ C \mid \tau_1 \to \tau_2 \mid X$ tipi

  $\sigma \in TS ::= \tau \mid \forall X.\sigma$ schemi di tipo

  Si osservi che il tipo polimorfo $(\forall X.X) \to Y$ non e' ne' un tipo ne' uno schema di tipo di ML
- $M \in \Lambda ::= x \mid c(\overline{M}) \mid \lambda x.M \mid M_1 M_2 \mid \mathrm{let}\ x = M_1\ \mathrm{in}\ M_2$

  Dal punto di vista della semantica operazionale il termine (let $x = M_1$ in M_2) di ML puo' essere identificato con $(\lambda x.M_2)M_1$. Tuttavia le regole per il type-checking di ML permettono di assegnare un tipo a let $x = M_1$ in M_2 anche quando $(\lambda x.M_2)M_1$ non e' tipabile.

  Dal punto di vista della semantica dei tipi di ML non vi e' niente da aggiungere rispetto a quanto visto in precedenza per i tipi polimorfi.
- Avendo distinto tra tipi e schemi di tipo, la definizione di typing judgement $\Gamma \vdash M\colon \tau$ richiede qualche precisazione. Supporremo che $\Gamma\colon Var \stackrel{fin}{\to} TS$ e $\tau \in T$, quindi le variabili nel typing context possono essere polimorfe, pero' ai termini assegnamo solo tipi. E' possibile considerare typing judgements $\Gamma \vdash M\colon \sigma$ che assegnato schemi di tipo a termini, pero' la versione piu' restrittiva permette di capire meglio l'estensione dell'algoritmo W

- Le regole di inferenza per derivare typing judgements senza informazioni di tipo nei termini sono

  (x) $\Gamma \vdash x{:}\,[S]\tau$ se $\forall\overline{X}.\tau = \Gamma(x)$ e $S{:}\,\overline{X} \to T$ (c) $\Gamma \vdash c{:}\,[S]\tau$ se $\forall\overline{X}.\tau = \sigma_c$ e $S{:}\,\overline{X} \to T$

  quando $\overline{X}$ e' vuoto si riottengono le regole usuali (x) e (c).

  $$(\to\mathrm{I})\ \frac{\Gamma, x{:}\,\tau_1 \vdash M{:}\,\tau_2}{\Gamma \vdash \lambda x{:}\,\tau_1.M{:}\,\tau_1 \to \tau_2} \qquad (\to\mathrm{E})\ \frac{\Gamma \vdash M_1{:}\,\tau_1 \to \tau_2 \quad \Gamma \vdash M_2{:}\,\tau_1}{\Gamma \vdash M_1 M_2{:}\,\tau_2}$$

  $$(\mathrm{let})\ \frac{\Gamma \vdash M_1{:}\,\tau_1 \quad \Gamma, x{:}\,\forall\overline{X}.\tau_1 \vdash M_2{:}\,\tau_2}{\Gamma \vdash \mathrm{let}\ x = M_1\ \mathrm{in}\ M_2{:}\,\tau_2}\ \overline{X} = \mathrm{FV}(\tau_1) - \mathrm{FV}(\Gamma)$$

  Si osservi che $\sigma_c \in TS$, quindi le costanti di ML possono avere un tipo polimorfo. La differenza essenziale tra (let $x = M_1$ in M_2) e $(\lambda x.M_2)M_1$ e' che la regola (let) permette di assegnare ad x uno schema di tipo, mentre la regola $(\to\mathrm{I})$ non lo consente.

L'algoritmo W di Hindley-Milner per ML va modificato per trattare i casi base x e c, e va' esteso per trattare il costrutto let-in. Le proprieta' dell'algortimo W precedentemente enunciate continuano a valere anche per l'estensione ad ML. Riportaimo solo i casi modificati e/o aggiunti.

- $W(\Gamma; x) = (Id_{\mathrm{FV}(\Gamma)}, [S]\tau)$ se $\Gamma(x) = \forall\overline{X}.\tau$ e $S{:}\,\overline{X} \hookrightarrow TVar$ ridenomina le $\overline{X}$ con variabili new.

  $W(\Gamma; c)$ e' definito in modo analogo usando $\sigma_c = \forall\overline{X}.\tau$.

- $W(\Gamma; \mathrm{let}\ x = M_1\ \mathrm{in}\ M_2) = ([S_2]S_1, \tau_2)$ se $(S_1, \tau_1) = W(\Gamma; M_1)$ e $(S_2, \tau_2) = W([S_1]\Gamma, x{:}\,\forall\overline{X}.\tau_1; M_2)$, dove $\overline{X} = \mathrm{FV}(\tau_1) - \mathrm{FV}([S_1]\Gamma)$.

Anche con l'estensione ad ML si ha che $W(\emptyset; \lambda x.xx)$ fallisce. Pero' adesso possiamo assegnare il tipo $X \to X$ a let $x = \lambda y.y$ in xx, poiche' alla variabile x e' assegnato lo schema di tipo $\forall X.X \to X$.

8.5 Correttezza in presenza di fix e Subject Reduction

In presenza di constanti per dimostrare la soundness delle regole di type-checking relativamente alla semantica dei tipi occorre far vedere che $c \in [\![\tau_c]\!]$. In genere cio' richiede qualche aggiustamento alla definizione di Λ (p.e. non considerando *cons* un termine ma un costruttore di termini $cons(M, N)$) o alla definizione di valore (p.e. considerando *cons* e *cons* V valori). Modulo questi aggiustamenti in genere non si presentano problemi ad estendere la dimostrazione di soundness. L'unica eccezione (tra gli esempi considerati) e' costituita dall'operatore di punto fisso fix. Infatti, se cerchiamo di dimostrare che $fix(\lambda x.M) \in X$ quando $(\lambda x.M) \in X \to X$ ci si accorge che avremo bisogno di assumere che cio' che vogliamo dimostrare (questo e' immediato quando si considera $fix(\lambda x.x)$. Un modo per evitare il problema e' considerare una interpretazione piu' restrittiva dei tipi, in cui $[\![\tau]\!]$ deve essere un *ideale*, un sottoinsieme di V_0 con proprieta' addizionali.

Se si vuole solo dimostrare la correttezza, si puo' evitare di considerare la semantica dei tipi (e la dimostrazione di soundness). La proprieta' che si deve dimostrare va sotto il nome di **Subject Reduction** (SR): $\vdash M{:}\,\tau$ e $M \Downarrow R$ implicano $\vdash R{:}\,\tau$ (e quindi $R \not\equiv err$. Tuttavia la demostrazione richiede una ipotesi induttiva (SR)+ piu' forte.

- dato $\rho{:}\,Var \overset{fin}{\to} \mathsf{Val}_0$ e $\Gamma{:}\,Var \to T$ (per semplicita' supponiamo di non avere variabili di tipo) diciamo che "$\rho{:}\,\Gamma$ vale" sse $\vdash \rho(x){:}\,\Gamma(x)$ vale per ogni $x \in dom(\Gamma)$.
- Ipotesi induttiva (SR)+: se $\Gamma \vdash M{:}\,\tau$, $\rho{:}\,\Gamma$ e $([\rho]M) \Downarrow R$ allora $\vdash R{:}\,\tau$.
- (SR)+ si puo' dimostra per induzione sulla derivazione di $([\rho]M) \Downarrow R$, case analysis sulla derivazione di $\Gamma \vdash M{:}\,\tau$), e sfruttando alcune regole ammissibili sui typing judgements, p.e. (sub).