SEAL: a language for self adaptive agents

Marco Tinacci

14 maggio 2012

Indice

1 Syntax

S	$::=$	$m(e, \ldots, e) \quad \mid \quad S \parallel_A S$	(System)
γ	$::=$	$\beta[a] \Rightarrow \delta \quad \mid \quad \gamma, \gamma$	(Rule)
β	$::=$	$\exists\, x : m\ .\ (\beta) \quad \mid \quad e \bowtie e \quad \mid \quad \mathbf{tt} \quad \mid \quad \beta \vee \beta \quad \mid \quad \neg\beta$	(Condition)
δ	$::=$	$< e > \alpha \quad \mid \quad \delta \oplus \delta$	(Distribution)
α	$::=$	$x = e; \quad \mid \quad \mathbf{noaction}; \quad \mid \quad \alpha\ \alpha$	(Action)

Sono state omesse le descrizioni di alcuni simboli per alleggerire la lettura della sintassi:

- m: riferimento alla definizione di un modulo,
- A: insieme di azioni di sincronizzazione,
- e: espressione,
- $\bowtie$: operatore di confronto,
- x: dichiarazione di un identificatore,
- a: label azione.

L'insieme dei moduli sarà quindi del tipo

$$M \subseteq \gamma \times \mathbb{VAR} \times \cdots \times \mathbb{VAR}$$

1.1 Syntactic sugar

Inseriamo un costrutto tale da poter inserire una condizione di abilitazione sugli elementi del supporto della distribuzione:

$$\beta[a] \Rightarrow < e, \beta' > \alpha \oplus \delta \equiv \beta \wedge \beta'[a] \Rightarrow < e > \alpha \oplus \delta, \beta \wedge \neg\beta'[a] \Rightarrow \delta$$

Introduciamo anche la visuale soggettiva del singolo modulo:

$$[m]_V \;||_A\; m_1 \;||_A\; \ldots \;||_A\; m_n \equiv m \;||_A\; V(m_1) \;||_A\; \ldots \;||_A\; V(m_n)$$

dove $V : M \to M$ è una funzione che mappa moduli in moduli.

2 Semantic

Andiamo a dare un'introduzione informale alla semantica del linguaggio descritto:

- *System*: un sistema può essere definito tramite un singolo modulo (m è un riferimento alla definizione di un modulo) o attraverso la composizione parallela di più sistemi su di un insieme di azioni di sincronizzazione $A \subseteq Act$;
- *Rule*: un insieme di regole definisce il comportamento di un modulo, se vale la condizione β si può passare alla valutazione della distribuzione δ;
- *Condition*: descrive una condizione fornendo anche un operatore di quantificatore esistenziale sui moduli;
- *Distribution*: descrive una distribuzione probabilistica di azioni α, dove ogni azione è accompagnata da un'espressione e, che ne descrive il peso, e un'azione a;
- *Action*: un azione descrive un aggiornamento dello stato, che può consistere nell'assegnamento di una, nessuna, o più variabili.

Definiamo la semantica del linguaggio in termini di *Markov Decision Processes*. Alla definizione di ogni modulo sarà assegnata una *MDP* della forma:

$$(\Sigma, Act, \to_\rho, \sigma_0)$$

dove

- $\Sigma = \{\sigma | \sigma : \mathbb{VAR} \to \mathbb{VAL}\}$ è l'insieme degli *stati* rappresentati da funzioni che mappano variabili in valori,
- Act l'insieme delle azioni,
- $\to_\rho \subseteq \Sigma \times Act \times Dist(U)$ è la relazione di *avanzamento* di stato,
- $\sigma_0 \in \Sigma$ è lo *stato iniziale*,

- $\rho \subseteq \beta \times Act \times Dist(U)$ è la *struttura statica* del *MDP*,
- $U = \{u | u : \Sigma \to \Sigma\}$ è l'insieme delle funzioni *update* di aggiornamento di stato.

$$\frac{(g, a, d) \in \rho}{\sigma \xrightarrow{a}_{\rho} d(\sigma)} \; \sigma \models g \quad \text{(Update)}$$

$$\rho_m = \{(g, a, d) \mid \gamma_m = g[a] \Rightarrow < e_1 > \alpha_1 \oplus \cdots \oplus < e_n > \alpha_n, d = [u_{\alpha_{i1}} : p_1, \ldots, u_{\alpha_{in}} : p_n]\}$$

dove $u_\alpha \in U$ è una funzione *update* definita nel seguente modo

$$u_\alpha(\sigma) = \begin{cases} \sigma[eval(e)/x] & \text{se } \alpha = x = e; \\ \sigma & \text{se } \alpha = \textbf{noaction}; \\ u_{\alpha''}(u_{\alpha'}(\sigma)) & \text{se } \alpha = \alpha'\alpha'' \end{cases}$$

Le probabilità sono invece calcolate nel seguente modo

$$p_i = \frac{eval(e_i)}{\sum_{j=1}^{n} eval(e_j)}, i = 1, \ldots, n$$

Salendo dal livello dei moduli a quello dei sistemi, introduciamo $\Pi \in Dist(S)$ per indicare distribuzioni di sistemi. Il sistema sarà rappresentato dal *MDP* risultante dal parallelo dei *MDP* che lo compongono.

$$\boxed{\begin{array}{lll} \dfrac{\sigma_m \xrightarrow{a}_{\rho_m} d(\sigma_m)}{S \xrightarrow{a} \Pi} & m \in S & \text{(Update)} \\[2ex] \dfrac{S_1 \xrightarrow{a} \Pi_1 \quad S_2 \xrightarrow{a} \Pi_2}{S_1 \; ||_A \; S_2 \xrightarrow{a} \Pi_1 \; ||_A \; \Pi_2} & a \in A & \text{(Sync)} \\[2ex] \dfrac{S_1 \xrightarrow{a} \Pi_1}{S_1 \; ||_A \; S_2 \xrightarrow{a} \Pi_1 \; ||_A \; S_2} & a \notin A & \text{(Async 1)} \\[2ex] \dfrac{S_2 \xrightarrow{a} \Pi_2}{S_1 \; ||_A \; S_2 \xrightarrow{a} S_1 \; ||_A \; \Pi_2} & a \notin A & \text{(Async 2)} \end{array}}$$

Rimane da definire come si comporta l'operatore di composizione parallela tra un sistema e una distribuzione e tra due distribuzioni.

$$\Pi_1 \; ||_A \; S_2(S) = \begin{cases} \Pi_1(S_1') & \text{se } S = S_1' \; ||_A \; S_2 \\ 0 & \text{altrimenti} \end{cases}$$

$$S_1 \; ||_A \; \Pi_2(S) = \begin{cases} \Pi_2(S_2') & \text{se } S = S_1 \; ||_A \; S_2' \\ 0 & \text{altrimenti} \end{cases}$$

$$\Pi_1 \; ||_A \; \Pi_2(S) = \begin{cases} \Pi_1(S_1) \cdot \Pi_2(S_2) & \text{se } S = S_1 \; ||_A \; S_2 \\ 0 & \text{altrimenti} \end{cases}$$

3 Examples

Esempio di un modulo di robot che esegue una *random walk* su una griglia escludedo dalla scelta probabilistica le direzioni adiacenti occupate:

$$
\begin{array}{lll}
m_1() & \triangleq & \mathbf{tt}[step] \Rightarrow \\
 & & < 1, \neg \; \exists \; m_1 \; v : v.x = x \wedge v.y = y + 1 > y = y + 1; \oplus \\
 & & < 1, \neg \; \exists \; m_1 \; v : v.x = x \wedge v.y = y - 1 > y = y - 1; \oplus \\
 & & < 1, \neg \; \exists \; m_1 \; v : v.x = x + 1 \wedge v.y = y > x = x + 1; \oplus \\
 & & < 1, \neg \; \exists \; m_1 \; v : v.x = x - 1 \wedge v.y = y > x = x - 1; \oplus \\
 & & < 1, \mathbf{tt} > \mathbf{noaction};
\end{array}
$$

Esempio di un modulo di robot analogo al precedente con la differenza che la scelta della mossa viene fatta in modo nondeterministico:

$$
\begin{array}{llllll}
m_2() & \triangleq & \neg \; \exists \; m_1 \; v : (v.x = x \wedge v.y = y + 1) & [north] & \Rightarrow & < 1 > y = y + 1; \\
 & & \neg \; \exists \; m_1 \; v : (v.x = x \wedge v.y = y - 1) & [south] & \Rightarrow & < 1 > y = y - 1; \\
 & & \neg \; \exists \; m_1 \; v : (v.x = x + 1 \wedge v.y = y) & [east] & \Rightarrow & < 1 > x = x + 1; \\
 & & \neg \; \exists \; m_1 \; v : (v.x = x - 1 \wedge v.y = y) & [west] & \Rightarrow & < 1 > x = x - 1; \\
 & & \mathbf{tt} & [stay] & \Rightarrow & < 1 > \mathbf{noaction};
\end{array}
$$