SEMWER 2021/22 – Zadanie domowe nr 1, am418402

Skoro wyrażenia arytmetyczne i logiczne mają standardowe znaczenie, to zakładam, że nie trzeba przytaczać ich semantyki w rozwiązaniu, zresztą byłaby taka sama jak na slajdach do wykładu. Skorzystam natomiast z możliwości rozszerzenia składni instrukcji i dodam instrukcję `endatomic`, która posłuży kończeniu bloków atomowych (wyjaśnię to niżej). Zdefiniujmy tymczasem najpierw zbiór konfiguracji semantyki programów:

$$\Gamma_p = (Prog \times State \times \mathbb{N}) \cup (State \times \mathbb{N})$$

i zbiór konfiguracji końcowych:

$$T_p = (State \times \mathbb{N})$$

Można zauważyć, że w porównaniu z semantyką TINY z wykładu, krotki konfiguracji zawierają dodatkową liczbę naturalną. Jej zadaniem będzie zarządzanie współbieżnością programu. Intuicyjnie dla programu $P = I_1||I_2||...||I_n$ działa ona trochę jak semafor znany z teorii współbieżności (i będę ją później nazywał zmienną semaforową): ma wartość 0, gdy żaden wątek nie wykonuje instrukcji z bloku `atomic`, a wartość k, gdy robi to wątek I_k (żadne dwa wątki nie mogą wykonywać instrukcji z bloku `atomic` jednocześnie) i wtedy będę mówił że "k-ty wątek ma semafor". Oczywiście chcemy, żeby wartością zmiennej semaforowej na początku wykonania programu było 0. Poza konfiguracjami semantyki programów będą potrzebne konfiguracje semantyki instrukcji:

$$\Gamma_i = (Instr \times State \times \mathbb{N}) \cup (Instr \times State \times \mathbb{N} \times ST) \cup (State \times \mathbb{N})$$

i znowu zbiór konfiguracji końcowych:

$$T_i = (State \times \mathbb{N})$$

Tym razem dorzucona liczba naturalna oznacza poziom zagnieżdżenia bloków `atomic` w danym wątku w danym momencie i w konfiguracji początkowej chcemy, żeby miała ona wartość 0. Można też zauważyć nowy element: ST (od skip-token). Jest to żeton (token) służący do kontroli wykonywania skoków po napotkaniu instrukcji `escape` - niżej wyjaśnię bardziej szczegółowo, jak powinien działać. Z grubsza chodzi o to, że jeśli jest obecny w krotce obecnie wykonywanej instrukcji, to instrukcja powinna zostać "pominięta", tj. nie zmienić stanu programu. Potrzebne teraz będą dwie relacje przejścia: programów oraz instrukcji. Będę pisał $K_1 \Rightarrow_p K_2$ gdy krotka K_1 jest w relacji przejścia programów z K_2 i analogicznie $K_1 \Rightarrow_i K_2$ gdy odpowiednie krotki są w relacji przejścia instrukcji. Zajmijmy się najpierw semantyką programów. Będziemy chcieli zapewnić prawidłowe, w szczególności atomowe wykonywanie instrukcji poszczególnych wątków. Niech więc relacja przejścia programów to najmniejsza taka relacja że po pierwsze

$$\langle (I_1||I_2||...||I_k||...||I_n), s, m_1 \rangle \Rightarrow_p \langle (I_1||I_2||...||I'_k||...||I_n), s', m_1 \rangle$$

wtedy i tylko wtedy gdy zachodzi ($m_1 \in \{0, k\}$) a ponadto przynajmniej jeden z warunków:

$$(\langle I_k, s, m_2, \rangle \Rightarrow_i \langle I'_k, s', m'_2, \rangle)$$

$$(\langle I_k, s, m_2, \rangle \Rightarrow_i \langle I'_k, s', m'_2, ST \rangle)$$
$$(\langle I_k, s, m_2, \rangle \Rightarrow_i \langle I'_k, s', m'_2, ST \rangle)$$
$$(\langle I_k, s, m_2, ST \rangle \Rightarrow_i \langle I'_k, s', m'_2, ST \rangle)$$

i dodatkowo wymagamy, żeby zbiór $\{m_1, m_2\}$ nie był równy $\{0, 1\}$. Ta reguła zapewnia, że k-ty wątek może wykonać swoją instrukcję wtedy kiedy "ma semafor" (czyli zmienna semaforowa ma wartość k) albo kiedy nikt tego semafora nie ma (czyli zmienna semaforowa ma wartość 0). Gdyby krotki z relacji instrukcji nie mogły zawierać dodatkowego żetonu ST, to wystarczyłoby podać pierwszy z powyższych warunków; niestety tak nie jest, więc musimy tu pracowicie uwzględnić różne postaci krotek. Natomiast dodatkowe wymaganie na zbiór $\{m_1, m_2\}$ wyklucza nam z tej reguły obsługę sytuacji, kiedy trzeba zmienić wartość zmiennej semaforowej - na to mamy osobne przypadki poniżej. Po drugie niech

$$(\langle I_1||I_2||...||I_k||...||I_n), s, m_1 \rangle \Rightarrow_p \langle (I_1||I_2||...||I_{k-1}||I_{k+1}||...||I_n), s', 0 \rangle$$

wtedy i tylko wtedy gdy zachodzi ($m_1 \in \{0, k\}$) oraz następujący warunek:

$$(\langle I_k, s, m_2 \rangle \Rightarrow_i \langle s', m'_2 \rangle)$$
$$(\langle I_k, s, m_2, SK \rangle \Rightarrow_i \langle s', m'_2 \rangle)$$

Ta reguła odpowiada za "znikanie" wątków po zakończeniu ich działania oraz oddawanie przez nie wtedy semafora. Tym razem nie uwzględniamy krotek zawierających żeton po prawej stronie, bo nie są one konfiguracjami końcowymi wątków (poniżej zapewnimy, że nie jest osiągalna taka sytuacja, że w krotce jest żeton ST, a nie ma pozostałej do wykonania instrukcji).
Po trzecie niech

$$\langle (I_1||I_2||...||I_k||...||I_n), s, 0 \rangle \Rightarrow_p \langle (I_1||I_2||...||I'_k||...||I_n), s', k \rangle$$

wtedy i tylko wtedy gdy:

$$(\langle I_k, s, 0 \rangle \Rightarrow_i \langle I'_k, s', 1 \rangle)$$

czyli to jest reguła odpowiedzialna za przyznanie danemu wątkowi semafora, kiedy wchodzi on na pierwszy poziom zagnieżdżenia. Krotek z żetonem ST nie musimy uwzględniać, bo w momencie wejścia do bloku `atomic` na pewno nie jesteśmy w fazie pomijania instrukcji, a więc tego żetonu nie ma. Wreszcie po czwarte:

$$\langle (I_1||I_2||...||I_k||...||I_n), s, k \rangle \Rightarrow_p \langle (I_1||I_2||...||I'_k||...||I_n), s', 0 \rangle$$

wtedy i tylko wtedy gdy zachodzi $m_1 \in \{0, k\}$ oraz jeden z warunków:

$$(\langle I_k, s, 1 \rangle \Rightarrow_i \langle I'_k, s', 0 \rangle)$$
$$(\langle I_k, s, 1, SK \rangle \Rightarrow_i \langle I'_k, s', 0 \rangle)$$

Ta reguła odpowiada za przywracanie zmiennej semaforowej na 0 ("zwalnianie" semafora) kiedy wątek, który miał semafor, wraca na zerowy poziom zagnieżdżenia.

Możemy teraz zająć się relacją przejścia semantyki instrukcji. Chcemy żeby to była najmniejsza taka relacja, dla której zachodzą następujące reguły:

$$\langle skip, s, m\rangle \Rightarrow_i \langle s, m\rangle$$

$$\langle x := e, s, m\rangle \Rightarrow_i \langle s[x \mapsto (\mathcal{E}[\![e]\!]s)], m\rangle$$

$$\langle I_1; I_2, s, m\rangle \Rightarrow_i \langle I_2, s', m'\rangle \quad \text{if} \quad \langle I_1, s, m\rangle \Rightarrow_i \langle s', m'\rangle$$

$$\langle I_1; I_2, s, m\rangle \Rightarrow_i \langle I_1'; I_2, s', m'\rangle \quad \text{if} \quad \langle I_1, s, m\rangle \Rightarrow_i \langle I_1', s', m'\rangle$$

$$\langle \text{ if } b \text{ then } I_1 \text{ else } I_2, s, m\rangle \Rightarrow_i \langle I_1, s, m\rangle \quad \text{if} \quad \mathcal{B}[\![b]\!]s = tt$$

$$\langle \text{ if } b \text{ then } I_1 \text{ else } I_2, s, m\rangle \Rightarrow_i \langle I_2, s, m\rangle \quad \text{if} \quad \mathcal{B}[\![b]\!]s = ff$$

$$\langle \text{ while } b \text{ do } I_1, s, m\rangle \Rightarrow_i \langle I_1; \text{ while } b \text{ do } I_1, s, m\rangle \quad \text{if} \quad \mathcal{B}[\![b]\!]s = tt$$

$$\langle \text{ while } b \text{ do } I_1, s, m\rangle \Rightarrow_i \langle s, m\rangle \quad \text{if} \quad \mathcal{B}[\![b]\!]s = ff$$

Dotychczasowe reguły są dość standardowe, ale to oczywiście jeszcze nie koniec:

$$\langle \texttt{atomic}\{I_1\}, s, m\rangle \Rightarrow_i \langle I_1; \texttt{endatomic}, s, m+1\rangle$$

Napotkawszy blok `atomic`, zwiększamy zagnieżdżenie o jeden, zostawiamy do wykonania ciało bloku a za nim umieszczamy znacznik końca bloku. Zdefiniowane wyżej reguły semantyki programów zapewnią atomowość wykonania bloku.

$$\langle \texttt{escape}, s, 0\rangle \Rightarrow_i \langle s, 0\rangle$$

`escape` na zerowym poziomie zagnieżdżenia miało być równoważne `skip`.

$$\langle \texttt{escape}, s, m\rangle \Rightarrow_i \langle s, m, ST\rangle \quad \text{if} \quad m > 0$$

Napotkawszy `escape` na niezerowym poziomie zagnieżdżenia, zostawiamy żeton ST na znak, że należy nie wykonywać instrukcji do następnego znacznika `endatomic` (a to jest znacznik pozostawiony przez najbardziej zagnieżdżony obecnie wykonywany blok atomowy - żadne nowe się nie pojawią po drodze, bo nie będziemy rozwijać bloków atomowych).

$$\langle I_1; I_2, s, m\rangle \Rightarrow_i \langle I_2, s', m', ST\rangle \quad \text{if} \quad \langle I_1, s, m\rangle \Rightarrow_i \langle s', m', ST\rangle$$

$$\langle I_1; I_2, s, m, ST\rangle \Rightarrow_i \langle I_2, s', m', ST\rangle \quad \text{if} \quad \langle I_1, s, m, ST\rangle \Rightarrow_i \langle s', m', ST\rangle$$

$$\langle I_1; I_2, s, m\rangle \Rightarrow_i \langle I_1'; I_2, s', m', ST\rangle \quad \text{if} \quad \langle I_1, s, m, \rangle \Rightarrow_i \langle I_1', s', m', ST\rangle$$

$$\langle I_1; I_2, s, m, ST\rangle \Rightarrow_i \langle I_1'; I_2, s', m', ST\rangle \quad \text{if} \quad \langle I_1, s, m, ST\rangle \Rightarrow_i \langle I_1', s', m', ST\rangle$$

Jak już zostawiliśmy żeton, to musimy zapewnić, że będzie się "propagował" w instrukcjach złożonych - po to powyższe cztery reguły.

$$\langle \texttt{endatomic}, s, m\rangle \Rightarrow_i \langle s, m-1\rangle$$

Jeśli nie ma żetonu ST, to `endatomic` tylko zmniejsza poziom zagnieżdżenia.

$$\langle \texttt{endatomic}, s, m, ST\rangle \Rightarrow_i \langle s, m-1\rangle$$

Jeśli żeton jest, to `endatomic` zabiera żeton ST z powrotem i zmniejsza poziom zagnieżdżenia.

$$\langle I_1; I_2, s, m, ST\rangle \Rightarrow_i \langle I_2, s', m'\rangle \quad \text{if} \quad \langle I_1, s, m, ST\rangle \Rightarrow_i \langle s', m'\rangle$$

$$\langle I_1; I_2, s, m, ST\rangle \Rightarrow_i \langle I_1'; I_2, s', m'\rangle \quad \text{if} \quad \langle I_1, s, m, ST\rangle \Rightarrow_i \langle I_1' s', m'\rangle$$

Tym razem musieliśmy zapewnić, żeby "propagował" się brak żetonu ST po zabraniu go przez `endatomic`. Pozostaje na koniec dodać reguły odpowiedzialne za to, żeby pozostałe instrukcje nie zmieniały stanu programu, jeśli obecny jest żeton ST (ale też nie "zdejmowały" żetonu):

$$\langle \mathit{skip}, s, m, ST\rangle \Rightarrow_i \langle s, m, ST\rangle$$

$$\langle x := e, s, m, ST\rangle \Rightarrow_i \langle s, m, ST\rangle$$

$$\langle \text{ if } b \text{ then } I_1 \text{ else } I_2, s, m, ST\rangle \Rightarrow_i \langle s, m, ST\rangle$$

$$\langle \text{ while } b \text{ do } I_1, s, m, ST\rangle \Rightarrow_i \langle s, m, ST\rangle$$

$$\langle \texttt{atomic}\{I_1\}, s, m, ST\rangle \Rightarrow_i \langle s, m, ST\rangle$$

$$\langle \texttt{escape}, s, m, ST\rangle \Rightarrow_i \langle s, m, ST\rangle$$