Komunikacja w CSP

Maciej Szklarczyk Jan Malisz Łukasz Nośko

2019-05-18

Spis treści

Rozdział 1

Wstęp

1.1 Początki CSP

CSP czyli Communicating Sequential Processes po raz pierwszy zostało opisane w 1978 przez Sir Charles Antony Richard Hoare'a. Należy pamiętać, że CSP nie jest językiem programowania - jest notacją. Nie posiada swojego kompilatora i środowiska do uruchamiania programów napisanych w CSP . Jednak należy wspomnieć, że CSP jako notacja stało się pierwowzorem dla pełnoprawnych języków programowania takich jak na przykład Occama czy GoLang.

1.2 CSP jako notacja synchroniczna

CSP jest notacją synchroniczną co oznacza, że komunikacja zachodzi zawsze pomiędzy dwoma procesami z których rozróżniamy dwa typy. Nadawcę i odbiorcę. Nadawca musi znać identyfikator odbiorcy i jawnie określić do kogo jest skierowana wiadomość. Odbiorca również musi być świadomy od kogo chce odebrać dane, aby mogło dojść do wymiany. Taką wymianę informacji nazywamy synchroniczną, ponieważ oba procesy o sobie wiedzą i znają swoje identyfikatory. Procesy jednak muszę oczekiwać na siebie, co prowadzi do tego, że odbiorca informacji będzie wstrzymany w stanie oczekującym dopóki nie otrzyma wiadomości. Analogicznie nadawca będzie oczekiwał na odbiorcę. Do komunikacji dojdzie kiedy obie strony przyjmą stan pozwalający na komunikację.

1.3 Niedeterminizm

Każdy projekt napisany w CSP jest zbiorem konkretnych deklaracji i instrukcji. Każda z tych instrukcji może wykonać się poprawnie bądź spowodować błąd. Jednak poprzez zdefiniowanie CSP jako notacji, a nie jako języka programowania wystąpienie błędu nie jest precyzyjnie zdefiniowane. Dodatkowo przez występujący niedeterminizm, niektóre instrukcje mogą powodować różne wyniki. Zatem dwa wykonania tego samego programu mogą zwrócić różne wyniki.

Rozdział 2

Składnia i semantyka

2.1 Instrukcje

Instrukcje najogólniejsze pojęcie w notacji CSP . Dzielą się one na deklaracje, instrukcje proste (instrukacja pusta, przypisanie, instrukcja wejścia i wyjścia) i strukrutalne. Tak jak było to już wspomniane, każda instrukcja w CSP wykonuje się poprawnie lub kończy się błędem.

2.2 Instrukcja pusta

Jest to, jak łatwo można się domyślić instrukcja która nie robi nic. Zawsze jej rezultatem jest powodzenie. Przydatna jest w instrukcjach alternatywy i iteracji. Zapisywana jest w sposób zaprezentowany na 2.1.

$$\text{skip} \tag{2.1}$$

2.3 Deklaracja

Oznacza stworzenie zmiennej przechowującej wartość o danym typie. Na 2.2 deklarowana jest zmienna o nazwie x i wartości typu integer.

$$x : integer \tag{2.2}$$

2.4 Przypisanie

W jej wyniku zmiannej docelowej po lewej stronie ustawiana jest wartość zwracana przez wyrażenie po prawej. Na 2.3 do zmiannej x przypisywana jest wartość 1.

$$x := 1 \tag{2.3}$$

2.5 Założenie sekwencyjne

Jest to kilka deklaracji lub instrukcji połączonych ze sobą za pomocą znaku średnika ';'. Wykonanie kolejnych elementów złożenia nasępuje kolejno po sobie,

gdy tylko poprzednie zakończy swoje działania z powodzeniem. W przypadku wystąpienia w złożeniu deklaracji, jej widoczność będzie sięgała do końca tego złożenia. W przykładzie 2.4 zaprezentowane jest złożenie które wykonuje podmianę zmiannych dzięki zmiennej tymczasowej.

$$\begin{aligned}&[x, y : integer;\\ &\ x := 1; y := 2;\\ &\qquad [tmp : integer;\\ &\qquad\ pom := x; x := y; y := pom]]\end{aligned} \tag{2.4}$$

2.6 Alternatywa

Alternatywa w CSP jest instrukcją strukturalną. W CSP jest niedeterministyczna. Składa się z oddzielonych znakiem pionowej kreski '|' instrukcji. Każda instrukcja, nazywana dozorowaną w najprostszej postaci składa się właśnie z dozoru, który jest wyrażeniem logicznym. Alternatywę rozpoczynamy od otwarcia kwadratowego nawiasu '[' a kończymy zamknięciem ']'. Wykonanie pojedynczej alternatywy polega na wyliczeniu wartości logicznej dozorów i w przypadku ich poprawności wykonanie instrukcji. W przypadku kiedy kilka dozorów jest poprawnych zostaje wykonana losowa instrukcja. Jeśli w przypadku poprawności jednego dozoru nie chcemy wykonać nic, używamy instrukcji 'skip'. Należy zadbać, żeby zawsze przynajmniej jeden dozor był prawdziwy. Przykład 2.5 prezentuje zastosowanie instrukcji pustej w alternatywie.

$$\begin{aligned}&[x < 0 -> y := -y\\ &|x \geq 0 -> \text{skip}]\end{aligned} \tag{2.5}$$

2.7 Dozory złożone

CSP pozwala na umieszczenie w jednej instrukcji wielu dozorów. Dozór na przykadzie 2.6 nazywany jest dozorem złożonym, ponieważ jego wykonanie polega na kolejnym wyliczaniu poszczególnych warunków logicznych. Jeśli dozór zwróci false, cała instrukcja jest zaniechana. W przypadku powodzenia, następuje przejście do następnego wyrażenia logicznego. Jeśli wszystkie wyrażenia są poprawne to cały dozór zostaje uznany za poprawny.

$$\begin{aligned}&[x > 10; x < 20 \rightarrow y := 1\\ &|x \leq 10 \rightarrow y := 2\\ &|x \geq 20 \rightarrow y := 3]\end{aligned} \tag{2.6}$$

Bibliografia

[1] Programowanie współbieżne i rozproszone/PWR Wykład 5 - Studia Informatyczne http://wazniak.mimuw.edu.pl/index.php?title=Programowanie_wsp%C3%B3%C5%82bie%C5