Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

AUTOREFERAT ROZPRAWY DOKTORSKIEJ

MGR INŻ. PIOTR MATYASIK

MODELOWANIE I ANALIZA SYSTEMÓW WBUDOWANYCH Z ZASTOSOWANIEM ALGEBRY PROCESÓW XCCS

PROMOTOR: dr hab. Marcin Szpyrka

Kraków, 14.10.2009

W rozprawie przedstawiono algebraiczno-graficzny język modelowania XCCS oparty na algebrach procesów CCS. Język ten, będący rozszerzeniem algebr CCS i pozwalający automatycznie uzyskiwać równoważne modele zapisane w algebrze CCS, został opracowany przede wszystkim do modelowania i weryfikacji dynamiki systemów wbudowanych. Poza opisem algebry CCS, zarówno w wersji elementarnej jak i rozszerzonej (z przesyłaniem danych) oraz algebry XCCS, w pracy podjęto próbę pokazania praktycznego aspektu zastosowania wprowadzonego języka modelowania. Opisano zaimplementowane narzędzia wspierające modelowanie z użyciem XCCS oraz pokazano przykład zastosowania XCCS do modelowania sterowania robotem kroczącym Hexor II.

Cel badań i teza pracy

Jako podstawowy cel podjętych badań przyjęto opracowanie rozszerzenia algebry procesów CCS, do formalizmu wyposażonego w graficzny język modelowania, który z jednej strony ułatwiałby konstruowanie modelu, a z drugiej eliminowałby problemy modelowania czysto algebraicznego.

Nowy język modelowania określony jako XCCS (eXtended CCS) miał być z założenia językiem algebraiczno-graficznym. W warstwie algebraicznej postanowiono pozostawić operatory powiązane z opisem dynamiki pojedynczych agentów, natomiast do warstwy graficznej postanowiono przenieść operatory używane przy definiowaniu połączeń między agentami. Jako punkt wyjścia do opracowania reprezentacji graficznej postanowiono przyjąć opisane w literaturze grafy przepływów. Notacja ta miała być jednak modyfikowana tak, by możliwe było jednoznaczne reprezentowanie wszystkich operatorów, które miały być przeniesione do warstwy graficznej.

Jako kolejny warunek odnoszący się do rozwijanego formalizmu przyjęto założenie o możliwości konwersji modeli zapisanych z użyciem XCCS do zapisu czysto algebraicznego w algebrze CCS (lub odpowiednio VP CCS). Oznaczało to konieczność opracowania algorytmów konwersji z modelu graficzno-algebraicznego do modelu algebraicznego. Podejście takie umożliwia skorzystanie z istniejących narzędzi do analizy modeli zapisywanych w algebrze CCS.

Przyjęto również założenie, że wraz z opracowaniem koncepcji XCCS podjęte zostaną prace związane z zaprojektowaniem i implementacją narzędzi komputerowych wspierających stosowanie rozwijanego formalizmu. Rozwijane oprogramowanie miało nie tylko zawierać edytor do konstruowania warstwy graficznej, ale również implementacje zdefiniowanych algorytmów konwersji. Podejście takie miało na celu dostarczenie jednocześnie opisu nowego języka modelowania wraz z oprogramowaniem umożliwiającym weryfikację jego przydatności.

Podsumowując, przyjęte tezy pracy można ściśle sformułować następująco:

Możliwe jest opracowanie algebraiczno-graficznego języka modelowania bazującego na algebrach procesów CCS oraz VP CCS, który:

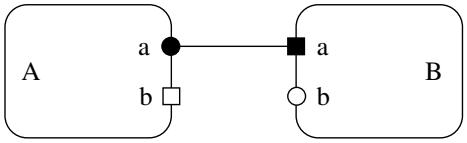
- *będzie umożliwiał wizualne modelowanie struktury połączeń między agentami tworzącymi konstruowany system;*
- *będzie eliminował możliwość wystąpienia błędów modelowania dotyczących połączeń między agentami, które pojawiają się w modelowaniu z użyciem języka czysto algebraicznego;*
- *wsparty przez odpowiednie oprogramowanie będzie umożliwiał uzyskanie równoważnego modelu w języku CCS (lub odpowiednio VP CCS) w sposób automatyczny.*

Algebra XCCS

XCCS ([4, 3]) jest graficznym rozszerzeniem algebry CCS. Warstwa graficzna pozwala na łatwiejsze konstruowanie i modyfikowanie modelu. Wzbogacenie algebry CCS o możliwości wizualnego projektowania zmniejsza liczbę błędów popełnianych w trakcie tworzenia modelu i pozwala na wielokrotne wykorzystanie raz skonstruowanych komponentów w różnych konfiguracjach. XCCS przenosi część operatorów algebry CCS do warstwy graficznej stąd formuły algebraiczne opisujące dynamikę agentów XCCS są uproszczone. Jest to kolejny element ułatwiający uczenie się modelowania z wykorzystaniem tego języka. Algebra XCCS posiada zintegrowane środowisko Inez służące do modelowania. Środowisko to pozwala również na automatyczną konwersję modelu do języka CCS w formacie zgodnym z CWB, w którym to narzędziu możne dokonać formalnej weryfikacji zaprojektowanego modelu.

Graficzna warstwa XCCS przyjmuje formę grafu nieskierowanego. Agent reprezentowany jest jako owal z nazwą w środku. Porty wejściowe agenta reprezentują okręgi, a kwadraty przedstawiają pory wyjściowe. Z każdym portem skojarzona jest jego etykieta, która znajduje się bezpośrednio przy nim. Restrykcje reprezentowane są poprzez zaczernienie odpowiednich portów.

Połączenia między portami należącymi do agentów reprezentowane są przez łączące je linie. Rysunek 1 przedstawia połączenie między portem a należącym do agenta A i portem a należącym do agenta B. Zbieżność nazw portów jest w tym przykładzie przypadkowa i nie warunkuje wystąpienia połączenia w modelu. Warunkiem wystąpienia połączenia w modelu XCCS jest jego obecność w warstwie graficznej. Podobnie nie ma połączenia między portami b w prezentowanym przykładzie, pomimo że nazwy są komplementarne.



Rys. 1. Graficzna reprezentacja połączenia portów w XCCS

Warstwa tekstowa XCCS służy do definiowania zachowania poszczególnych agentów. Do budowania wyrażeń można użyć ograniczonego zbioru operatorów dostępnych w CCS oraz TCCS: *prefiks* (.), *wybór* (+), *silny wybór* (++), *opóźnienie* (\$), *przeplot* (?). Operatory prefiksu (.), wyboru (+),

silnego wyboru (++) i opóźnienia ($) zachowują się tak jak w języku CCS ([2]). Natomiast operator przeplotu (?) generuje sumę wszystkich możliwych kolejności akcji wyszczególnionych etykiet.

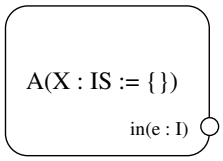
Prawidłowy model może być transformowany do czystego skryptu CCS. Algorytm automatycznej konwersji został opracowany i zaimplementowany w przedstawionym dalej edytorze Inez.

Algebra XCSS z przesyłaniem danych

Algebra XCCS z przesyłaniem danych (VP-XCCS) jest rozszerzeniem algebry XCCS o możliwość parametryzowania agentów oraz etykiet. Algebra ta nie wprowadza większych możliwości w kwestii modelowania niż XCCS, ale umożliwia bardziej zwięzły, skondensowany zapis modelu. Analogicznie jak w przypadku algebry XCCS, która automatycznie konwertowana jest do skryptu CCS, tak VP-XCCS konwertowana jest do VP-CCS, a następnie z użyciem narzędzia *jvp* ([1]) do rachunku CCS. Algorytm automatycznej konwersji został również zaimplementowany w edytorze Inez.

Z VP-XCCS celowo usunięto pewne konstrukcje dostępne w VP-CCS w celu poprawienia przejrzystości modelu i przystosowania narzędzia do praktyki inżynierskiej. Wprowadzone restrykcje w bardzo niewielkim stopniu ograniczają możliwości modelowania, przy jednoczesnej poprawie krzywej uczenia dla proponowanego rozwiązania.

Podobnie jak w przypadku wersji XCCS wykorzystującej rachunek CCS bez przesyłania danych, również w VP-XCCS agent reprezentowany jest jako owal z nazwą w środku. Nie ulegają także modyfikacji symbole i oznaczenia portów przyporządkowanych do agenta. Podobnie jak poprzednio kwadrat reprezentuje port wyjściowy, a okrąg port wejściowy. Port w kolorze czarnym oznacza port przeznaczony do komunikacji wewnętrznej. Dodatkowe informacje pojawiają się natomiast w nazwach portów oraz w nazwie agenta (zob. rys. 2).



Rys. 2. Graficzna reprezentacja agenta VP-XCCS

Nazwa agenta może zostać wzbogacona o jego stan początkowy. Jest on wtedy bezpośrednio widoczny w warstwie graficznej i jest to preferowany sposób przedstawiania tej własności. Innym sposobem przekazania stanu początkowego agenta jest przeniesienie tej informacji do warstwy tekstowej. W takim przypadku nazwa agenta jest identyczna jak w przypadku modeli w XCCS, a w warstwie tekstowej może ona przyjąć następującą formę:

```
agent A = A(X : IS := {})
agent A(X : IS := {})  =  ...
```

W rachunku VP-XCCS, w porównaniu do XCCS, nazwa portu w agencie może zostać rozszerzona o nazwę parametru i jego typ. Elementy te umieszcza się w nawiasach okrągłych za nazwą portu. Wymagane jest deklarowanie wszystkich używanych typów danych, a ich definicje mają charakter globalny dla modelu. Przykład sparametryzowanego portu pokazano na rys. 2. Przez port wejściowy `in` przesyłany jest parametr `e` typu `I`.

Warstwa algebraiczna dla algebry VP-XCCS składa z dwóch zasadniczych części. Pierwsza z nich zawiera definicje typów i stałych używanych w modelu. Druga część to równania definiujące agentów, w których można stosować wcześniej zdefiniowane typy i stałe. Tekstowa definicja agenta jest ściśle związana z jego graficzną reprezentacją.

W szczególności wymagane jest aby:

- nazwa agenta w warstwie graficznej pokrywała się z nazwą pierwszego agenta występującego w definicji algebraicznej (tzw. *nazwa podstawowa*) wraz z ewentualnymi parametrami,

- wszystkie porty użyte w definicji algebraicznej muszą pojawić się w warstwie graficznej wraz z ewentualnymi argumentami i typami i na odwrót.

Definicja agenta rozpoczyna się od słowa kluczowego **`agent`**, po którym występuje nazwa agenta z ewentualnymi argumentami. Następnie musi wystąpić znak równości, a po nim wyrażenie definiujące agenta. Warstwa tekstowa definicji agenta VP-XCCS może składać się z definicji wielu wzajemnie powiązanych ze sobą agentów. W definicji pojedynczego agenta można użyć następujących operatorów: sekwencji (`.`), sumy (+), silnej sumy (++), przeplatania (`?`), opóźnienia (`$`). Ponadto dostępne jest wyrażenie warunkowe, które pozwala na wybór jednego z agentów w zależności od podanego warunku logicznego. Przy konstruowaniu wyrażenia warunkowego można stosować operatory logiczne (**`not`**, **`and`**, **`or`**), operatory arytmetyczne (+, −, ⋆, **`div`**, **`mod`**) oraz operatory relacyjne (=, <>, <, <=, >=, >). VP-XCCS nie udostępnia wszystkich konstrukcji dostępnych w VP-CCS obsługiwanym przez konwerter *jvp*. Celem tych ograniczeń jest uproszczenie procesu modelowania, skrócenie czasu potrzebnego do opanowania narzędzia oraz uniknięcie nieintuicyjnych konstrukcji mogących powodować późniejsze problemy.

Konwersja diagramów VP-XCCS w znacznej części oparta jest na algorytmie wykorzystywanym dla wersji bez przesyłania danych.

Inez XCCS Editor

Inez (rys. 3) jest narzędziem CAD wspomagającym modelowanie systemów współbieżnych z wykorzystaniem języka XCCS. Składa się ono z graficznego edytora do edycji diagramów XCCS oraz zestawu wbudowanych narzędzi konwertujących diagramy odpowiednio do skryptów CCS lub VP-CCS. Inez jest wolnym oprogramowaniem rozpowszechnianym na licencji GPL.

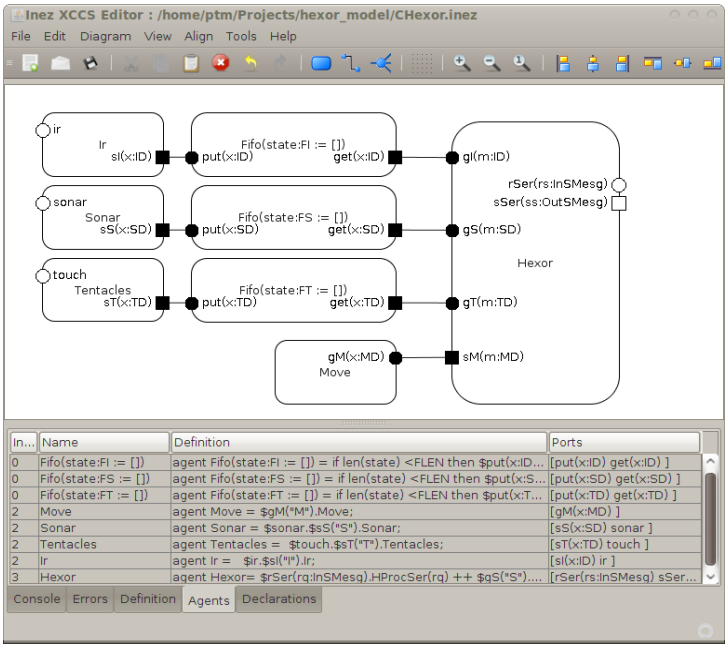
Edytor został napisany w języku *Java* z wykorzystaniem środowiska programistycznego *Netbeans*. Sam komponent renderujący diagramy XCCS został stworzony na bazie biblioteki *JGraph*. Posiada ona dużą ilość gotowych procedur służących do operowania na grafach, począwszy od podstawowych prymitywów jak: węzły, krawędzie, porty, poprzez możliwość etykietowania poszczególnych elementów, a kończąc na ułatwieniach typu: siatki, wielopoziomowy mechanizm „cofnij/ponów” i zaawansowanej serializacji. Użycie tego narzędzia znacznie przyspieszyło utworzenie graficznej reprezentacji diagramów wraz z wszystkimi potrzebnymi procedurami służącymi do edycji. Reszta graficznego interfejsu użytkownika opiera się na standardowych komponentach biblioteki *Swing*.

Wszystkie parsery kodu pracujące w edytorze zostały wykonane przy użyciu generatora parserów *AntLR* (ANother Tool for Language Recognition) wraz z IDE je wspierającym (*AntLRWorks*). Jest to zestaw narzędzi pozwalających na tworzenie skanerów, parserów, interpreterów, kompilatorów i translatorów w oparciu o ich gramatyczny opis. Poza językiem *Java*, *AntLR* wspiera również języki C, C++, Ada95, Perl, JavaScript i inne.

Modelowanie systemów wbudowanych z wykorzystaniem VP-XCCS na przykładzie robota Hexor

Podsumowaniem prac jest prezentacja procesu modelowania systemu sterującego robota Hexor a konkretnie wbudowanego systemu mikroprocesorowego, którego zadaniem jest sterowanie niskopoziomowe ([5]). W skład zadań tego systemu wchodzi sterowanie serwami, odczytywanie czujników, komunikacja z systemem nadrzędnym oraz minimalna, *zachowawcza inteligencja*. W pracy zaprezentowano konstrukcję robota, koncepcję niskopoziomowego systemu sterującego, model formalny wykonany w języku VP-XCCS, jego formalną weryfikację oraz wybrane fragmenty implementacji w języku C.

Ze względu na pewne ograniczenia jakie posiada oryginalne oprogramowanie dostarczane z robotem *Hexor*, opracowano alternatywną koncepcję systemu sterującego warstwą niskopoziomową na-



Rys. 3. Inez XCCS editor – edycja modelu robota Hexor

zwaną *HexorNG*. Wykonano model, który odpowiadał opracowanej strukturze systemu sterującego. Finalnie na tak skonstruowanym modelu przeprowadzono testy zgodności ze specyfikacją. Sygnałami wejściowymi dla modelowanego układu są dane z czujników: dotykowego, podczerwonego, ultradźwiękowego oraz wiadomości z portu szeregowego. Wyprowadzanie danych odbywa się tylko poprzez port szeregowy.

Opracowany model robota Hexor został skonwertowany za pomocą edytora *Inez* oraz aplikacji *jvp* do formatu akceptowanego przez CWB. W tej aplikacji przeprowadzona została weryfikacja modelowa opracowanego systemu.

Weryfikacja modelu oparta jest na funkcji `checkproperty` wbudowanej w program CWB. Funkcja ta sprawdza czy podany system spełnia formułę logiki Hennessy-Milnera. Stąd na potrzeby weryfikacji utworzone zostały formuły definiujące pożądane cechy modelu odpowiadające założonym wymaganiom odnośnie pracy sterownika. Dodatkowo system sterujący został przetestowany wbudowaną komendą szukającą zakleszczeń oraz sprawdzony został rozmiar przestrzeni stanów badanego modelu. Wynik obliczeń pozwalał stwierdzić, że system zachowuje się zgodnie z założeniami.

Implementacja systemu sterującego została wykonana przy użyciu zestawu narzędzi *avr-gcc* dostępnego na licencji GPL. Jako środowisko programistyczne został wykorzystany *Eclipse* wraz z wtyczką wspomagającą tworzenie oprogramowania dla architektury AVR. Zadania w systemie mikroprocesorowym zbudowane są na bazie API dostarczanego przez AvrX mikrokernel. Podobnie komunikacja między zadaniami jest realizowana w oparciu o implementację kolejek wiadomości dostarczaną przez ten system.

Podsumowanie

W rozprawie przedstawiono graficzne rozszerzenie jednego z bardziej popularnych rachunków algebraicznych CCS. Proponowane podejście nie tylko stanowi propozycję wprowadzenia wizualnego modelowania do algebr CCS, ale również znacznie ułatwia ich stosowanie, eliminując możliwość wystąpienia niektórych rodzajów błędów pojawiających się podczas modelowania z użyciem algebr CCS. Nie bez znaczenia pozostaje fakt, że proponowany w rozprawie język modelowania XCCS jest kompatybilny z algebrami CCS. W zależności od użytej wersji języka (z przesyłaniem danych lub bez), automatycznie można uzyskać równoważny model zapisany w elementarnym rachunku CCS

lub w rachunku z przesyłaniem danych (VP CCS). Niezależnie od przypadku, w efekcie końcowym można zastosować dostępne narzędzia do weryfikacji modeli zapisanych w algebrach CCS (pakiet CWB). Podejście takie pozwala na praktyczne stosowanie algebry XCCS już na opisanym etapie rozwoju, bez konieczności rozwijania własnych narzędzi do weryfikacji modeli.

W rozprawie nie skupiono się wyłącznie na zaprezentowaniu teoretycznego opisu proponowanego języka modelowania. Istotna część prezentowanych rozwiązań ma charakter praktyczny. Wykonano projekt i zaimplementowano narzędzia do wizualnego modelowania z użyciem algebry XCCS (edytor *Inez*). Zaimplementowane zostały również algorytmy konwersji, co pozwala łatwo zweryfikować przydatność proponowanego rozwiązania. W ramach rozprawy zaprezentowano również praktyczny przykład zastosowania proponowanego formalizmu. Użyto go do zamodelowania systemu sterowania robotem Hexor II. Opisany w pracy materiał pozwala na ocenę XCCS zarówno od strony teoretycznej jak i praktycznej.

Podsumowując powyższe rozważania, do istotnych osiągnięć pracy należy zaliczyć:

- opracowanie algebraiczno-graficznego języka modelowania stanowiącego rozszerzenie algebr procesów CCS w kierunku modelowania wizualnego, zarówno w wersji dla rachunku elementarnego, jak i rozszerzonego (z przesyłaniem danych);
- opracowanie algorytmu konwersji modeli XCCS do modeli zapisywanych w algebrze CCS (w formacie zgodnym z CWB), dla modeli bez przesyłania danych;
- opracowanie algorytmu konwersji modeli XCCS do modeli zapisywanych w algebrze VP CCS (w formacie zgodnym z *jvp*), dla modeli z przesyłaniem danych;
- zaprojektowanie i implementacja narzędzi wspierających projektowanie systemów wbudowanych z użyciem formalizmu XCCS (edytor *Inez*) wraz z implementacją opracowanych algorytmów konwersji;
- pokazanie zastosowania języka modelowania XCCS na przykładzie systemu sterowania robotem kroczącym Hexor II.

Wszystkie przedstawione powyżej wnioski prowadzą do konkluzji, iż wyposażona w graficzny język modelowania algebra procesów XCCS wraz ze wspierającym ją oprogramowaniem może być przydatnym narzędziem w procesie wytwarzania oprogramowania dla systemów wbudowanych. Jednocześnie stanowi to potwierdzenie tez postawionych w rozprawie.

Literatura

[1] G. Bruns. A language for value-passing CCS. Internal Report ECS-LFCS-91-175, University of Edinburgh, aug 1991.

[2] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[3] M. Szpyrka and P. Matyasik. Formal modelling and verification of concurrent systems with XCCS. In *Proc. of the 7th International Symposium on Parallel and Distributed Computing*, Kraków, Poland, July 1-5 2008.

[4] M. Szpyrka and P. Matyasik. Graphical modelling tool for ccs process algebra. In *Software engineering techniques in progress*, page 81–94, Brno, Czech Republic, 2008.

[5] M. Szpyrka and P. Matyasik. Modelling hexor robot behaviour with value passing xccs. In *Software engineering techniques in progress*, page 67–78, Kraków, 2009. Wydawnictwa AGH.