Modele i języki do specyfikacji naukowych przepływów prac operujących kolekcjami danych

Autoreferat rozprawy doktorskiej

Jacek Sroka

Instytut Informatyki
Uniwersytet Warszawski

16 października 2008

1 Wprowadzenie

Przeżywające dynamiczny rozwój nauki stosowane stwarzają nieustanne zapotrzebowanie na poszerzanie istniejącej wiedzy informatycznej oraz rozwój obecnie stosowanych technik komputerowych. Nowe wyzwania stawiane są między innymi przed bazami danych oraz przepływami prac. Jest tak dlatego, że przetwarzane są coraz większe zbiory danych, a procesy analityczne stosowane do tego stają się coraz bardziej skomplikowane. Na styku tych dwóch dziedzin wyrósł nowy interesujący obszar badań.

W wielu naukach stosowanych, jak ekologia, geologia, chemia, astronomia i szczególnie bioinformatyka, strukturalne dane są przetwarzane przez system zorganizowany w rodzaj sieci, przez którą przepływają i są przetwarzane dane. Krawędzie sieci wyznaczają kierunek przepływu danych, a węzły wykonują stosowne operacje. Taka organizacja eksperymentów obliczeniowych przypomina przepływy prac stosowane w biznesie, jednak tutaj więcej uwagi poświęca się przetwarzaniu kolekcji danych, a mniej zaawansowanym zagadnieniom kontroli sterowania. Z tego powodu proponujemy termin *naukowe przepływy danych* (NPD, *ang. Collection-Oriented Scientific Workflows*).

Podstawowe operacje w NPD to typowe dla danej dziedziny zastosowań algorytmy operujące na danych. Ich wydajne implementacje są najczęściej rozpowszechniane jako otwarte oprogramowanie lub udostępniane przez instytucje badawcze na publicznie dostępnych serwerach Internetowych. Wyniki obliczeń wykonywanych przez NPD są podstawą do formułowania naukowych hipotez oraz do ich uzasadniania lub obalania. Sposób organizacji NPD, to znaczy, rodzaj wykonywanych przez nie operacji podstawowych oraz zależ-

ności między tymi operacjami, jest zazwyczaj publikowany w jakiejś postaci razem z wynikami eksperymentu. Jest to niezbędne, by recenzenci oraz inni badacze zainteresowani eksperymentem mogli zrozumieć na czym on dokładnie polega, potrafili efektywnie i obiektywnie ocenić jego wartość, powtórzyć i zweryfikować go, a nawet zaadoptować na własne potrzeby.

Tradycyjnie takie eksperymenty obliczeniowe były wykonywane przez kopiowanie i wklejanie danych do lokalnie uruchamianych programów, np. komponentów pakietu EMBOSS [21], oraz do formularzy WWW serwerów obliczeniowych dostępnych w Internecie, np. FASTA Sequence Comparison na Uniwersytecie w Virginii [24] lub Basic Local Alignment Search Tool w NCBI [16]. Taka metoda eksperymentowania była żmudna i podatna na błędy. Czasami wspomagano się też pisanymi *ad hoc* skryptami i programami, które automatyzowały pewne zadania. Jednak wymagało to przynajmniej podstawowych umiejętności programistycznych oraz znajomości zagadnień programowania rozproszonego. Co więcej większość wytworzonego oprogramowania była nieprzenośna oraz posiadała bardzo skąpą dokumentację lub nie posiadała jej w ogóle. W praktyce w wypadku tak przeprowadzonego eksperymentu do publikacji nadawały się jedynie wyniki końcowe, a sam NPD był w najlepszym razie opisywany w języku naturalnym.

Obecnie do definiowania i wykonywania eksperymentów NPD używane są specjalizowane naukowe środowiska badawcze jak Taverna [17, 13] lub Kepler [14]. Oparte są one na prostych lecz zarazem posiadających dużą siłę wyrazu notacjach graficznych. Integrują najważniejsze narzędzia, usługi internetowe oraz bazy danych z danej dziedziny, a także posiadają wiele dodatkowych funkcji jak np. śledzenie proweniencji (pochodzenia) danych bądź semantyczne odkrywanie nowych usług. Modele, języki i techniki stosowane przy modelowaniu NPD stały się tym samym potrzebnym obszarem badań i są głównym tematem tej rozprawy. Mimo, że jest to nowa dziedzina, wykazuje wiele podobieństw z modelowaniem przepływów prac, modelowanie procesów biznesowych, bazami danych, gridami obliczeniowymi oraz wieloma innymi uznanymi obszarami badań.

Rozprawa składa się z dwóch części. W pierwszej badamy formalną semantykę języka specyfikacji NPD zastosowanego w Tavernie, która jest jednym z najpopularniejszych narzędzi NPD używanych w bioinformatyce. Naszym celem było dokładne i wyczerpujące zdefiniowanie wszystkich cech tego języka, zbadanie ich przydatności oraz przedyskutowanie alternatyw. Mimo, że dążyliśmy do eleganckiej definicji, zgodziliśmy się jedynie na minimalną liczbę kompromisów, tak żeby opisać Tavernę najwierniej jak to możliwe. Dzięki

temu, gdy w przyszłości powstanie bardziej zwięzły model rdzenia funkcjonalności Taverny, będzie można oceniać na ile jest dokładny.

Ponieważ formalizacja okazała się zawiła, zaczęliśmy się zastanawiać czy nie jest możliwe opracowanie prostszego języka opisu NPD z łatwiejszą do zrozumienia semantyką. W tym celu zbadaliśmy czy i jak można wykorzystać wyniki na temat baz danych i modelowania klasycznych przepływów prac.

Istnieją już dobrze zbadane modele formalne przepływów prac jak sieci Petriego [15, 20] jednak w badaniach nad nimi nacisk kładziony jest na przepływ sterowania, a przetwarzanie zagnieżdżonych kolekcji danych nie jest podejmowane. Istnieją również dobrze poznane formalne modele operowania zagnieżdżonymi kolekcjami danych jak *nested relational calculus* (NRC) [4], które z drugiej strony ignorują zagadnienia przepływu sterowania. W ramach tej rozprawy podjęliśmy próbę opracowania formalnego modelu NPD, który łączyłby zarówno zagadnienia przepływu sterowania jak i operowania na danych. Model ten nazwaliśmy DataFlow Language (DFL). Naszym celem było przy tym pozostanie najbliżej jak to możliwe istniejących modeli z obu dziedzin, tak aby dało się dostosować dostępne dla nich wyniki teoretyczne. Jednocześnie badamy czy i w jaki sposób ten hybrydowy model może być przydatny w praktyce, to znaczy podczas prowadzenia rzeczywistych eksperymentów obliczeniowych. W tym celu zbudowaliśmy nowe narzędzie oparte na DFL i testowaliśmy je na rzeczywistych przykładach dostosowanych z systemu Taverna.

2 Formalna semantyka Taverny

W pierwszej części prac zbadaliśmy i przedyskutowaliśmy formalną semantykę Taverny. Pokazaliśmy również że zdefiniowana przez nas semantyka może być użyta do dowodzenia właściwości NPD wyrażonych w Tavernie. W tym celu wykazaliśmy, że NPD zdefiniowane w Tavernie posiadają własność terminacji.

Podstawową motywacją dla opracowania formalnej semantyki jest rosnąca popularność eksperymentów NPD wykonywanych za pomocą Taverny przy jednoczesnym wzroście ich skomplikowania. Tym samym pojawia się potrzeba opracowania automatycznych procedur weryfikacji, podobnych do tych stosowanych przy złożonych transakcjach biznesowych. Istnienie formalnej semantyki jest warunkiem wstępnym do podjęcia takich prac.

Inną dziedziną dla której formalna semantyka jest niezbędna jest opty-

malizacja wykonania. Tak samo jak w przypadku zapytań do baz danych, eksperymentator mógłby jedynie specyfikować co ma zostać obliczone, a za wybór najbardziej efektywnej strategii obliczeń odpowiadałby silnik narzędzia NPD. Dodatkowo ze względu na duże zainteresowanie NPD, w Internecie zaczęły powstawać repozytoria opisanych w ten sposób eksperymentów [10]. Wykonywanie zapytań na takich repozytoriach to ciekawy problem [5, 2]. Potencjalny język zapytań do repozytorium NPD powinien uwzględniać ich semantykę, a nie tylko składnię, to znaczy porównywać jak dane NPD działają, a nie tylko jak są zdefiniowane.

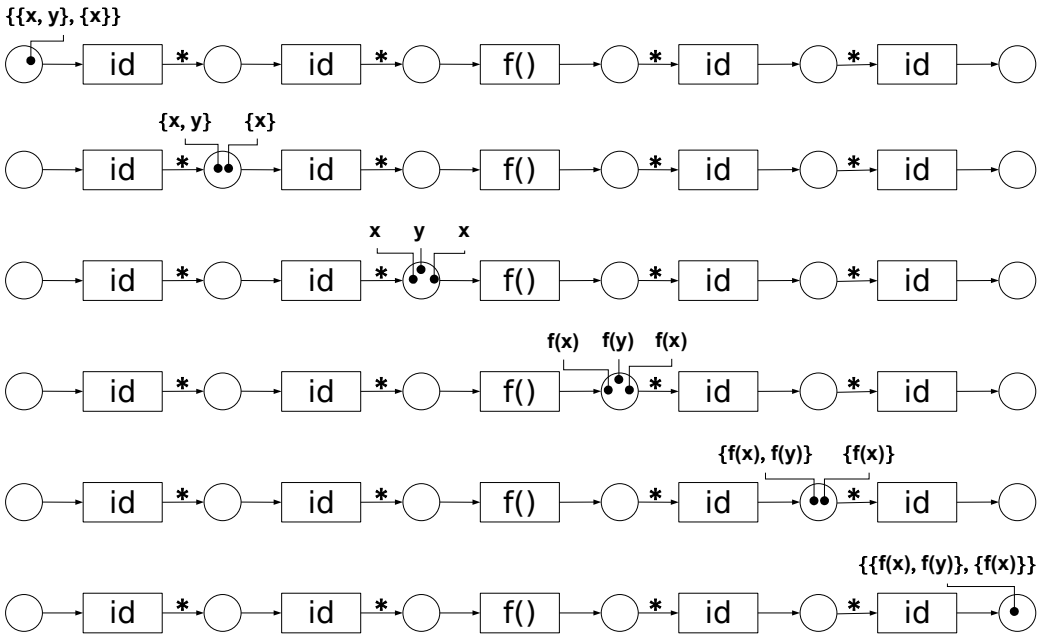
Uważamy również, że samo sformułowanie semantyki jest przydatne, ponieważ zmusza nas do pełnej i dokładnej analizy zachowania Taverny. Opracowanie eleganckiej i naturalnej semantyki jest dobrym testem czy zachowanie systemu jest spójne i właściwie dobrane. Jednocześnie, takie prace są inspiracją do zastanowienia się nad alternatywnymi definicjami niektórych elementów. Tym samym formalne opisanie semantyki może pomóc w pracach rozwojowych Taverny.

3 DFL

DFL powstał przez połączenie NRC i sieci Petriego. Z NRC wzięty został system typów oraz zestaw podstawowych operatorów. Dzięki temu powinno być możliwe zaadoptowanie technik optymalizacji znanych z baz danych. Organizacja przepływu sterowania opiera się na graficznej notacji sieci Petriego, w której dane przenoszone są po krawędziach sieci przez żetony oraz przetwarzane przez tranzycje. Wybór sieci Petriego jest umotywowany ich prostotą połączoną jednocześnie z dużą siłą wyrazu oraz dostępnością wielu wyników teoretycznych, np. dotyczących analizy poprawności.

Podstawową operacją NRC jest operator **map**, który przyjmuje zbiór i funkcję operującą na jego elementach i zwraca zbiór powstały przez obliczenie funkcji dla wszystkich argumentów z oryginalnego zbioru. Aby pozwolić na podobny rodzaj strukturalnej rekursji [23] w DFL wprowadziliśmy specjalny rodzaj krawędzi, które rozkładają zbiór na elementy i składają go z powrotem.

Prosty przykład pokazujący zagnieżdżoną iterację znajduje się na Rys. 1. Jeżeli w lewym miejscu umieścimy żeton przenoszący zbiór zbiorów, zostanie powielony przez tranzycję *id* i rozłożony na elementy. Następnie wynikowe żetony reprezentujące zbiory, które były elementami zbioru wejściowego są same rozkładane przez drugą parę operacji identyczności i krawędzi rozkła-



Rysunek 1: Przykład zagnieżdżonej iteracji

dającej zbiór. Dalej każdy element rozłożonych podzbiorów jest przetwarzany przez funkcję $f()$ a wynik jest dwukrotnie składany przez kolejne operacje identyczności i krawędzie składające. Aby zagwarantować, że elementy pochodzące z różnych zbiorów nie zostaną wymieszane podczas składania oraz że składanie zachodzi jedynie gdy wszystkie niezbędne żetony już dotarły, każdy żeton przenosi dodatkową informację o historii rozkładania swojego elementu. Jej omówienie wykracza poza zakres tego autoreferatu.

Jak pokazujemy w rozprawie mimo że istotnie rozszerzyliśmy sieci Petriego i wprowadziliśmy nowe problemy związane z przepływem sterowania określonym przez wartości danych, możliwe jest przeniesienie na DFL niektórych znanych dla nich wyników teoretycznych. W tym celu omawiamy sposób konstruowania specyfikacji NPD w DFLu, który gwarantuje, że zawsze będą spełniały pewne kryteria poprawności. Stosujemy w tym celu dobrze znaną technikę gdzie sieć jest krok po kroku i zgodnie z pewnymi regułami rozbudowywana przez zastąpienie jej tranzycji lub miejsc przez większe sieci. Zestawy podobnych reguł były badane przez Berthelota [3] i Muratę [15] jako redukcje zachowujące własności żywotności i ograniczoności sieci Petriego. Były również stosowane przez van der Aalsta [25], Reijersa [19] oraz

Chrząstowskiego-Wachtela [6] do generowania sieci przepływów prac.

Następnie dowodzimy, że NPD generowane zgodnie z naszymi regułami posiadają własność *semipoprawności* (*ang. semisoundness*). To znaczy prawdą jest że: (1) każde obliczenie daje dokładnie jeden wynik i po jego zwróceniu bezwzględnie się kończy, to znaczy po rozpoczęciu obliczeń z pojedynczym żetonem w miejscu wejściowym, gdy w miejscu wyjściowym pojawi się jakiś żeton, to nigdzie indziej nie ma już żadnych innych żetonów, oraz (2) nie ma obliczeń które nie mogą się zakończyć wyprodukowaniem wyniku, to znaczy z każdego stanu osiągalnego po rozpoczęciu obliczeń z pojedynczym żetonem w miejscu wejściowym można osiągnąć stan z pojedynczym żetonem w miejscu wyjściowym. Semipoprawność jest odpowiednikiem klasycznej poprawności (*ang. soundness*) sieci przepływów prac [1] dla modeli, w których decyzje co do przepływu sterowania mogą być determinowane przez wartości przesyłanych danych.

Przy okazji wyników teoretycznych prezentujemy również DFL designer — stworzone przez nas narzędzie NPD, które jest oparte na notacji DFL i oferuje wszystkie operacje bioinformatyczne z systemu Taverna oraz posiada kilka unikatowych wśród narzędzi NPD cech jak: analiza poprawności, transformacja eksperymentów do wyrażeń NRC w celu ich optymalizacji oraz wsparcie dla debugowania i testowania.

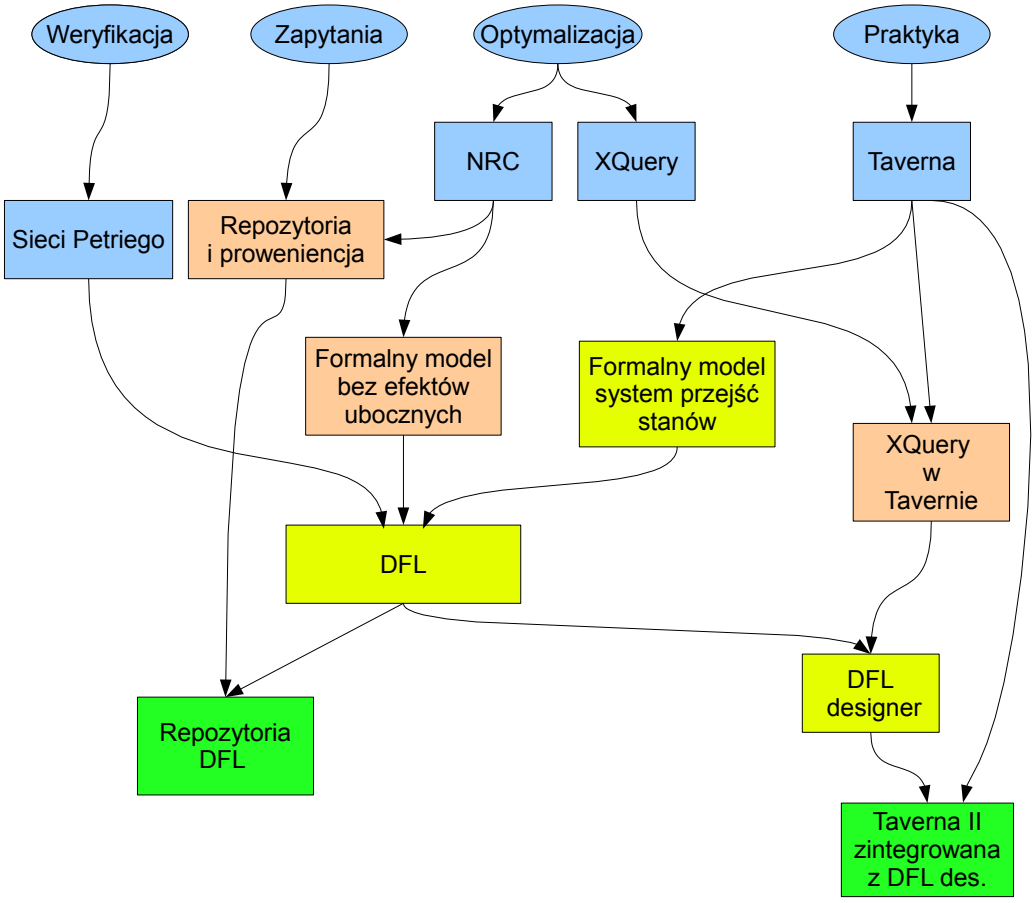
4 Podsumowanie opublikowanych wyników

Wyniki przedstawione w rozprawie zostały opublikowane lub są w trakcie publikacji w ramach kilku prac z kilkoma różnymi współautorami. Tu wskazujemy te prace.

Formalna semantyka Taverny została zgłoszona jako praca do czasopisma Fundamenta Informaticae. Materiał o DFLu, z wyłączeniem prezentacji narzędzia DFL designer, opublikowaliśmy w pracach [12, 8]. Praca przedstawiająca DFL designer jest w trakcie przygotowania.

Wyniki zebrane w rozprawie są częścią rozleglejszych badań na powiązane tematy, które również zaowocowały kilkoma publikacjami z naszym udziałem i do których odwołujemy się w tekście rozprawy. Na Rys. 2 przedstawiono związki między tymi pracami, a wynikami przedstawionymi w rozprawie. Niebieskie prostokąty reprezentują istniejące wyniki, ceglaste reprezentują nasze wyniki, do których odwoływaliśmy się w rozprawie, żółte reprezentują wyniki stanowiące główny wkład tej pracy, a zielone naturalne kierunki dalszych

badań. Strzałki przedstawiają wzajemny wpływ i przepływ pomysłów.



Rysunek 2: Współzależności i kontekst wyników przedstawionych w rozprawie

W ramach tych dodatkowych prac przyjrzeliśmy się na ile efekty uboczne wykonywanych operacji są istotne w NPD oraz pokazaliśmy, że jeżeli efekty uboczne nie występują, to do wyrażania NPD wystarczy sam NRC [9, 8]. Rozszerzyliśmy również Tavernę o możliwość wykonywania zapytań XQuery oraz specyfikowania przy ich pomocy części eksperymentu obliczeniowego [22]. W końcu opisaliśmy formalnie model repozytoriów NPD [11].

Dodatkowo zaproponowaliśmy oraz przetestowaliśmy w praktyce pomysł by do definiowania i wykonywania NPD użyć arkusza kalkulacyjnego. W tym celu rozszerzyliśmy [7] arkusz Calc z pakietu OpenOffice.org [18] o podstawową funkcjonalność Taverny.

Literatura

[1] W.M.P. van der Aalst. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[2] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 343–354. ACM, 2006.

[3] Gèrard Berthelot. Checking properties of nets using transformation. In *Advances in Petri Nets 1985, covers the 6th European Workshop on Applications and Theory in Petri Nets-selected papers*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40, London, UK, 1986. Springer-Verlag.

[4] Peter Buneman, Shamim Naqvi, Val Tannen, and Limsoon Wong. Principles of programming with complex objects and collection types. *Theoretical Computer Science*, 149(1):3–48, 1995.

[5] Vassilis Christophides, Richard Hull, and Akhil Kumar. Querying and splicing of XML workflows. In *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 386–402, London, UK, 2001. Springer-Verlag.

[6] Piotr Chrząstowski-Wachtel, Boualem Benatallah, Rachid Hamadi, Milton O'Dell, and Adi Susanto. A top-down Petri net-based approach for dynamic workflow modeling. In Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske, editors, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 336–353. Springer, 2003.

[7] Marek Dopiera, Adam Kawa, Piotr Krewski, Jacek Sroka, Jerzy Tyszkiewicz, and Tomek Weksej. Tavernalc: How to transform your OpenOffice Calc into a grid. In *OpenOffice.org Conference (OOoCon)*, 2007.

[8] Anna Gambin, Jan Hidders, Natalia Kwasnikowska, Sławomir Lasota, Jacek Sroka, Jerzy Tyszkiewicz, and Jan Van den Bussche. NRC as a

formal model for expressing bioinformatics workflows. Poster at ISMB, 2005. Poster.

[9] Anna Gambin, Jan Hidders, Natalia Kwasnikowska, Sławomir Lasota, Jacek Sroka, Jerzy Tyszkiewicz, and Jan Van den Bussche. Well-constructed workflows in bioinformatics. In *Workshop on Database Issues in Biological Databases (DBiBD)*, 2005.

[10] C. A. Goble and D. C. De Roure. myExperiment: social networking for workflow-using e-scientists. In *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pages 1–2, New York, NY, USA, 2007. ACM Press.

[11] Jan Hidders, Natalia Kwasnikowska, Jacek Sroka, Jerzy Tyszkiewicz, and Jan Van den Bussche. A formal model of dataflow repositories. In *Proc. of the 4th Int. Workshop on Data Integration in Life Sciences (DILS)*, volume 4544/2007 of *LNBI*, pages 105–121, Philadelphia, PA, USA, June 27–29 2007.

[12] Jan Hidders, Natalia Kwasnikowska, Jacek Sroka, Jerzy Tyszkiewicz, and Jan Van den Bussche. DFL: A dataflow language based on Petri nets and nested relational calculus. *Information Systems*, 33(3):261–284, 2008.

[13] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R. Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucl. Acids Res.*, 34:W729–732, 2006.

[14] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system: Research Articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[15] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[16] National Center for Biotechnology Information. NCBI Blast. `http://www.ncbi.nlm.nih.gov/blast/Blast.cgi`.

[17] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna:

a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, November 2004.

[18] OpenOffice.org — the free and open productivity suite. `http://www.openoffice.org`.

[19] Hajo A. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*. Number 2617 in Lecture Notes in Computer Science. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[20] Wolfgang Reisig. *Petri nets: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[21] P. Rice, I. Longden, and A. Bleasby. EMBOSS: the European Molecular Biology Open Software Suite. *Trends in Genetics*, 16(6):276–277, June 2000.

[22] Jacek Sroka, Grzegorz Kaczor, Jerzy Tyszkiewicz, and Andrzej M. Kierzek. XQTav: an XQuery processor for Taverna environment. *Bioinformatics*, 22(10):1280–1281, May 2006.

[23] Dan Suciu and Limsoon Wong. On two forms of structural recursion. In *ICDT*, pages 111–124, 1995.

[24] University of Virginia. FASTA Sequence Comparison. `http://wrpmg5c.bioch.virginia.edu/fasta_www2/fasta_list2.shtml`.

[25] Wil M. P. van der Aalst. Verification of workflow nets. In *ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pages 407–426, London, UK, 1997. Springer-Verlag.