Proposta de Colaboração CNPq-NSF
Protem-CC-CNPq

CLoP^N

Sistemas Escaláveis de Alto Desempenho para Programação Lógica com Restrições

Vítor Santos Costa, Inês de Castro Dutra e Felipe M. G. França
COPPE/Sistemas, UFRJ
http://www.cos.ufrj.br

Cláudio Geyer
Instituto de Informática, UFRGS
http://www.inf.ufrgs.br

Gopal Gupta e Enrico Pontelli
Laboratory for Logic, Databases and Advanced Programming, NMSU
http://www.cs.nmsu.edu/lldap/

Declaração: *Uma proposta correspondente de colaboração está sendo submetida pelos pesquisadores americanos mencionados nesta página: Gopal Gupta e Enrico Pontelli.*

Conteúdo

1 Equipe

1.1 No Brasil

- Investigador Principal: Vítor Santos Costa, COPPE/Sistemas, UFRJ
- Co-investigadores:
  - Inês de Castro Dutra e Felipe Maia Galvão França, COPPE/Sistemas, UFRJ
  - Claudio Geyer, Instituto de Informática, UFRGS
  - Jorge Barbosa e Cristiano Costa, Departamento de Computação, UCPEL
  - Juarez Assumpção Muylaert Filho, IPRJ-UERJ
  - Doris Ferraz de Aragon, Priscila Machado Vieira Lima, Fatima Dargam e Antonio Saraiva Branco, ILTC-UFF

1.2 Nos EUA

- Investigador Principal: Gopal Gupta, Department of Computer Science, New Mexico State University
- Co-investigador: Enrico Pontelli, Department of Computer Science, New Mexico State University

1.3 Colaboradores Externos

- Europa
  - Manuel Hermenegildo, Departamento de Inteligência Artificial, UPM, Espanha.
  - Fernando Silva, Faculdade de Ciências, LIACC, Universidade do Porto, Portugal.
  - Massimo De Gregorio e Ernesto Burattini, Istituto di Cibernetica-CNR, Nápolis, Itália
- EUA
  - David Scott Warren, Departamento de Computação, Universidade de Nova York em Stony Brook
  - Juliana Freire, Bell Labs

2 Resumo e Objetivos do Projeto

O projeto tem como objetivo principal o desenvolvimento de um conjunto de ferramentas para a *programação lógica com restrições* (*constraints*) de alto desempenho através do suporte de *paralelismo* e *hardware programável*.

Programação Lógica tem como base uma linguagem de programação de alto nível, baseada em lógica matemática, mais especificamente cláusulas de Horn. Programação Lógica com restrições introduz uma nova característica à linguagem, que permite ao programador expressar problemas por meio de restrições. O mecanismo de execução se encarrega de resolver o conjunto de equações/inequações expressas pelas restrições.

Programação com restrições [26, 55, 37] é um paradigma de programação que tem recebido considerável atenção recentemente, como é demonstrado pelo sucesso das conferências anuais sobre este tema (e.g, sessão inteira na IJCAI) e pelo aparecimento de um número considerável de livros e revistas científicas da área. O paradigma tem sido aplicado na resolução de classes

importantes de problemas em áreas tais como escalonamento (job-shop scheduling), alocação de recursos, bancos de dados, e problemas de otimização em geral. Diferentes companhias como ILOG (França), ALS (USA), Cosytec ou Siemens AG (Alemanha) fornecem soluções comerciais baseadas em restrições (vide sessão industrial ocorrida na última Joint International Conference and Symposium on Logic Programming, JICSLP'98). Como outro exemplo, recentemente o IC-Parc foi criado no Imperial College, UK, fundamentalmente para aplicar a tecnologia de restrições.

Sistemas que utilizam suporte para programação com restrições aplicam técnicas eficientes (forward checking, look ahead, revised simplex, etc) para resolver problemas que envolvem um espaço de busca muito grande. No entanto, mesmo com a aplicação destas técnicas, o espaço de busca pode ainda ser considerável. Consequentemente, programas com restrições podem rodar em tempo exponencial, sendo verificado que algumas das aplicações podem facilmente consumir dias de CPU em máquinas sequenciais de alto desempenho.

Neste contexto, nossa proposta é pesquisar técnicas de implementação eficientes para sistemas de programação lógica com restrições através do uso de *paralelismo* e *hardware programável*.

O suporte ao paralelismo surge como uma das formas principais de acelerar a execução dos programas. Em particular, tem-se notado nos últimos tempos o aparecimento de "clusters" de máquinas de memória compartilhada, tais como as máquinas Beowulf [42], projeto NOW [5] ou NCP2 [4, 9], que usam máquinas do tipo PC para construir sistemas altamente paralelos, mas de baixo custo. Aplicações da programação com restrições exibem considerável paralelismo, tanto entre diferentes sub-tarefas (paralelismo-E) [27] como na enumeração de diferentes alternativas (paralelismo-OU). Este paralelismo pode ser explorado tanto através de máquinas de memória compartilhada como de memória distribuída, e em particular, através de máquinas da classe Beowulf.

Uma técnica alternativa para a melhoria do desempenho é através da migração de operações de software para hardware programaável. Até recentemente, o custo de desenvolvimento de novo hardware era proibitivo para aplicações especializadas. O aparecimento de novas tecnologias de hardware, tais como as FPGAs (Field Programmable Gate Arrays), permitiu alterar esta equação, diminuindo dramaticamente os custos. Neste projeto, pretendemos obter ainda melhor performance do sistema através da identificação e posterior migração de partes críticas de código fonte Prolog-like para bancos locais de FPGAs.

Com a experiência adquirida com projetos e implementações passadas [13, 39, 29, 16, 52, 57, 50, 22, 24, 20, 21], pretendemos desenvolver um ambiente de programação que ofereça ao usuário uma linguagem de alto nível, declarativa, com ferramentas de suporte à depuração e visualização da execução paralela dos programas, além do suporte de software/hardware eficientes.

No contexto deste projeto temos como objetivos:

- Fornecer ao programador uma ferramenta de programação de alto nível, baseada em lógica de primeira ordem, robusta, eficiente para rodar aplicações práticas.
- Estudar técnicas de implementação de linguagens que utilizam restrições.
- Estudar técnicas eficientes de paralelização em sistemas com restrições.
- Estudar a potencialidade de máquinas paralelas modernas em processamento simbólico.
- Estudar algoritmos de escalonamento.
- Desenvolver aplicações em processamento simbólico.
- Desenvolver técnicas de compilação para o uso de hardware programável visando executar porções críticas de código fonte do tipo Prolog.

O sistema vai ser implementado em máquinas distribuídas, preferencialmente em rede, utilizando um hardware de baixo custo (Pentiums com dois ou quatro processadores). A máquina abstrata será desenvolvida com base nos sistemas ALS (comercial) e clp(FD) [19] (domínio público).

3 Resumo de Interações anteriores

Todos os pesquisadores envolvidos neste projeto já têm colaborado anteriormente em projetos comuns e todos têm trabalhado em áreas afins. O professor Gopal Gupta, co-diretor e co-fundador do laboratório de lógica, bancos de dados e programação avançada da NMSU, e o professor Enrico Pontelli, também da NMSU, contribuíram ativamente através do projeto APPELO (Protem 3-CC) com sua experiência na implementação de sistemas paralelos de programação lógica para máquinas de memória compartilhada. O prof. Gopal Gupta já veio ao Brasil em visita de colaboração. No projeto APPELO também colaboraram os professores Vítor Santos Costa, na época afiliado à Universidade do Porto, Inês de Castro Dutra (COPPE/Sistemas), Claudio Geyer (UFRGS), Jorge Barbosa e Cristiano Costa (UCPEL).

Os pesquisadores Felipe Maia Galvão França, Priscila Machado Vieira Lima, Inês de Castro Dutra, Massimo De Gregorio e Burattini também já colaboraram em outros projetos: NEUROCON (Protem 2-CC) e ICOM (Protem 3-CC). A ênfase dessa pesquisa é em mapeamento de lógica de primeira ordem/proposicional em redes neuronais artificiais (RNAs), e posterior implementação em FPGAs. Como resultado de uma colaboração dentro do projeto ICOM, foi possível desenvolver um programa para o qual está sendo requisitada uma licença de direitos autorais. Tal sistema visa a síntese automática de controladores inteligentes de sinais de trânsito em hardware programaável em FPGAs [23, 11]. A especificação das regras que descrevem o comportamento adequado de um sinal de trânsito alvo é feita em lógica proposicional, que é mapeada em redes neuronais e finalmente realizada em hardware programável.

O professor Juarez Muylaert tem uma vasta experiência em projeto e implementação de linguagens funcionais, no que diz respeito a interpretação abstrata de programas, compilação e implementação. Algumas técnicas utilizadas em linguagens funcionais são anteriores às utilizadas em linguagens lógicas e servem como base para a implementação de ambientes de programação em lógica. Os professores Felipe e Juarez estão colaborando no projeto de implementação de uma versão estruturada da linguagem GAMMA [25], cujos resultados são diretamente aplicáveis aos objetivos desse projeto.

A pesquisadora Juliana Freire trabalhou durante alguns anos com o professor D. S. Warren na SUNY, Stony Brook, na área de programação lógica com tabulação visando aplicações a bancos de dados [44]. Atualmente, Juliana está trabalhando no Bell Labs e desenvolve sua pesquisa em aplicações de programação lógica para bancos de dados e WWW. Seu interesse é em otimização de consultas na WEB (utilizando restrições) e dimensionamento de redes de chaveamento. O professor David Warren é um dos maiores especialistas mundiais em programação em lógica. Recentemente tem dado ênfase à implementação de sistemas de restrições para lógica com tabulação e suas aplicações. Existe colaboração prévia entre os professores David Scott Warren, Juliana Freire e Vítor Santos Costa através do projeto português PROLOPPE e do projeto luso-americano REAP.

O professor Fernando Silva tem vasta experiência na implementação de sistemas paralelos e distribuídos que exploram paralelismo E e OU [52, 43, 16], e tem colaborado com os professores Vítor Santos Costa, Inês Dutra e Gopal Gupta. O professor Fernando Silva esteve em visita ao Brasil através do projeto ALFA financiado pela comunidade européia.

A experiência e colaboração prévia entre todo o conjunto de pesquisadores é fator fundamental para o sucesso desse projeto. Reunimos pesquisadores que são especialistas nas áreas de sistemas paralelos de programação lógica, sistemas híbridos (lógico-conexionista), técnicas de compilação, programação de FPGAs, visando aplicações reais.

4 Atividades

O objetivo básico do projeto é obter máximo desempenho para aplicações reais em programação lógica e com restrições, em hardware de baixo custo (redes de PCs ou heterogêneas, com máquinas conectadas por uma rede fast ethernet ou Mirynet), onde porções críticas de código fonte (Prolog-like) possam ser implementadas em FPGAs.

Para isto dividimos o projeto nas seguintes atividades principais:

1. Implementação
   (a) Ambiente de execução
      i. Programação lógica com restrições (NMSU, ALS Inc., UFRJ, UFRGS, UPorto, Stony Brook);
      ii. Migração de trechos críticos para hardware (UFRJ, ILTC, Istituto di Cibernetica)
      iii. Paralelização do sistema (NMSU, UFRJ, UFRGS, UPorto)
      iv. Escalonamento (UFRJ, UFRGS, NMSU)
   (b) Ambiente de Compilação
      i. Compilação de Prolog com Restrições (NMSU, UFRJ, IPRJ)
      ii. Deteção de independência entre objetivos (UFRGS, NMSU, Madrid)
      iii. Análise de Granularidade (UFRGS, UFRJ, IPRJ)
      iv. Deteção de dependência entre objetivos (UFRGS, NMSU, Madrid)
      v. Geração de código especializado para o hardware FPGA (UFRJ, Istituto di Cibernetica, ILTC)
   (c) Ferramentas de visualização e depuração (UFRJ, NMSU)
2. Aplicações
   (a) Aplicações ALS, Inc. (NMSU, ALS Inc.)
   (b) Otimização de consultas na WEB (Bell Labs, UFRJ)
   (c) Dimensionamento de sistemas (Bell Labs, UFRJ)
3. Análise de Performance (UFRJ, NMSU)
4. Avaliação do Projeto (UFRJ, UFRGS, UCPEL, ILTC, IPRJ, NMSU)

O sistema base para o nosso trabalho será o DAOS [13], um sistema protótipo paralelo de programação lógica baseado na `wamcc` [15] e desenvolvido durante o projeto APPELO (Protem 3-CC). DAOS foi desenvolvido para sistemas distribuídos, utilizando técnicas de consistência relaxada para manter coerência de memória para estruturas de dados compartilhadas, ou seja, utiliza técnicas de software DSM (Distributed Shared Memory) [30, 31]. Troca de trabalho é feita através de troca de mensagens, de forma explícita para diminuir o overhead de comunicação. A utilização de técnicas de software DSM torna a implementação do sistema menos custosa, visto que o programador utiliza um modelo de programação baseado em memória compartilhada. DAOS utiliza uma série de otimizações que diminuem o overhead de manipulação das áreas compartilhadas.

4.1 Implementação

As tarefas de implementação podem ser organizadas segundo (a) Ambiente de Execução, (b) Compilação, e (c) Ferramentas, tais como de depuração e visualização gráfica.

4.1.1 Ambiente de Execução

No ambiente de execução consideraremos as questões de (i) implementação de restrições, (ii) migração para hardware, (iii) problemas de paralelização, e (iv) escalonamento de tarefas.

Técnicas de implementação de restrições em programação lógica Nosso projeto não pretende se concentrar no estudo de novas técnicas de implementação de restrições, tentaremos ao máximo utilizar tecnologia já existente, e em particular, software "open-sourced", isto é, para o qual os fontes estão livremente disponíveis. Dentro dos domínios a considerar, o mais importante é certamente o domínio finito, para o qual têm sido desenvolvidas as mais importantes aplicações. Outros domínios a considerar são os domínios dos racionais, dos reais, e dos booleanos.

Esta tarefa será dividida em duas: (i) estudo e 'porting' dos 'solvers' para o DAOS, (ii) paralelização dos 'solvers'. Na tarefa (i) será inicialmente considerado o 'solver' para domínios finitos, sendo usado como base de trabalho o sistema clp(FD) desenvolvido por Diaz e Codognet no INRIA, França [19].

Migração de trechos críticos para hardware Desenvolvimentos recentes na tecnologia de hardware permitem a geração e configuração dinâmica especializada em hardware programável a baixo custo. O objetivo desta tarefa é o de detectar fragmentos do programa que são particularmente importantes, avaliar o custo do mapeamento destes fragmentos e, se factível, configurar dinamicamente os circuitos correspondentes em hardware programável. Para tal serão aproveitados os trabalhos recentes na implementação de de lógica de primeira ordem sobre redes neuronais artificiais, que apresentam propriedades muito adequadas para reconfiguração dinâmica.

Esta tarefa pode ser dividida em:

- Especificação dos trechos críticos em lógica de primeira ordem

  Esta tarefa consiste em estudar o comportamento dos programas através de 'profiles' da execução, com a finalidade de identificar cláusulas no programa que são ou (i) muito frequentemente utilizadas na execução ou (ii) consomem muito tempo durante a execução.

- Mapeamento dos trechos críticos para redes neuronais

  Esta tarefa diz respeito à construção das redes neuronais correspondentes às partes de código que foram identificadas como sendo críticas no programa fonte. Serão investigadas algumas técnicas já existentes e utilizadas para mapeamento de lógica proposicional. Inicialmente, o mapeamento será de lógica proposicional para redes neuronais binárias [33, 32, 2, 3]. A seguir, serão investigados os custos de implementação dos modelos para mapeamento e execução de lógica de primeira ordem [33, 34].

- Migração dos trechos críticos para hardware

  Esta tarefa consiste na produção de circuitos digitais correspondentes ao comportamento dos diversos neurônios definidos na tarefa anterior e sua estrutura de interconexão. Um programa fonte VHDL (linguagem de descrição de hardware [54, 1]) é gerado e é a partir deste programa que o hardware programável (ou não) é configurado.

Paralelização do sistema E possível reconhecer diferentes formas de paralelismo num programa lógico com restrições (CLP). Paralelismo OU resulta da existência de diferentes soluções alternativas, que são procuradas por enumeração num sistema CLP. Paralelismo E pode ser classificado como independente e dependente. Paralelismo dependente resulta da natureza dinâmica da execução. Um exemplo é o paralelismo encontrado na avaliação simultânea de restrições, onde a computação é do tipo produtor-comsumidor. Paralelismo independente resulta da possibilidade de sub-divisão do problema em sub-problemas independentes.

A exploração das diversas formas de paralelismo neste contexto é reconhecidamente um problema complexo. Propostas tradicionais sofrem ou de custos elevados, ou são altamente restritivas quanto ao tipo de paralelismo a explorar. O trabalho dos proponentes em projetos anteriores permitiu definir um sistema para a execução distribuída de programas Prolog, DAOS, que permite explorar tanto paralelismo-OU como paralelismo independente-E, tendo sido estabelecido um modelo de execução e implementado um protótipo inicial. O sistema DAOS será a base para o trabalho na implementação do sistema paralelo a desenvolver dentro deste projeto. Esse trabalho consistirá nas seguintes sub-tarefas:

- Validação continuada dos modelos de execução. Em colaboração com os investigadores em NMSU será feita uma comparação com os diferentes modelos recentemente propostos: "sparse binding arrays" como originalmente proposta para DAOS, cópia preguiçosa como proposta em δCOWL [49], e "split-choicepoints" [28], propostos pelos nossos parceiros na NMSU. Todos estes métodos permitem suportar as várias formas de paralelismo de interesse e todos eles apresentam vantagens específicas.

- Implementação de restrições, isto é, partilha ou distribuição completa de "stores". Numa primeira fase, para um sistema distribuído, acreditamos que a a distribuição e cópia de "stores" é a alternativa mais interessante, e será usada na extensão de DAOS.

- Implementação de primitivas de controle para programação com restrições: muita da complexidade em programação com restrições resulta da necessidade de implementar heurísticas para controle do espaço de procura. Estas heurísticas serão fortemente afetadas pela execução paralela. O nosso objetivo é o de inicialmente suportar as primitivas existentes em sistemas sequenciais, e depois considerar novas primitivas para execução paralela.

Escalonamento A implementação de um sistema que explora paralelismo implícito como é o da proposta requer um esforço maior de programação paralela. Devido à natureza irregular das computações em sistemas paralelos de programação lógica é necessária a elaboração de algoritmos eficientes de distribuição de tarefas durante a execução [8, 20, 14, 18, 56]. Sistemas paralelos de programação lógica que exploram múltiplas formas de paralelismo exigem técnicas sofisticadas de escalonamento para distribuir tarefas E e OU que surgem dinamicamente durante a execução. Com a introdução de restrições no sistema tabulação no para resolver problemas do tipo quando executar restrições em paralelo.

4.1.2 Ambiente de Compilação

Programas declarativos fornecem substancial possibilidade de otimização. Essas otimizações tanto podem permitir o melhoramento do desempenho para execução sequencial, como o reconhecimento das oportunidades de paralelização existentes no programa. A deteção de oportunidades de paralelização é particularmente importante para paralelismo-E, e será a principal motivação deste projeto. O objetivo da tarefa consistirá portanto no desenho de um analisador global para programas declarativos, sendo focadas as técnicas de análise para programas lógicos com restrições.

Existe substancial trabalho sobre a análise global de programas lógicos (sem restrições). O trabalho é baseado na técnica de interpretação abstracta, um método geral de análise global proposto pelos Cousots para linguagens imperativas [17]. Este método foi inicialmente adaptado para linguagens funcionais por Mycroft e outros [40], e para linguagems de programação lógica por Mellish e outros [38]. No caso de linguagens de programação em lógica os resultados mais importantes têm sido obtidos através da abstração das semânticas operacionais dos programas, apesar de haver resultados interessantes para métodos baseados nas semânticas de ponto-fixo [36]. De particular interesse para o projeto destaca-se o trabalho de proponentes do projeto no uso de

análise global para a execução paralela de programas Prolog em Andorra-I [48], no estudo de independência entre objetivos e no estudo da granulosidade de objetivos [51, 35, 58, 7, 6].

A questão a resolver nesta tarefa coloca-se no desenvolvimento de técnicas globais para programas com restrições. A abstração proposta para programas Prolog não funciona para programas com restrições. De fato o conjunto de restrições opera como um contexto global que é transportada pelo programa, tornando particularmente difíceis de resolver questões como a independência de objetivos. As seguintes questões serão investigadas:

- Estudo comparativo de métodos de análise "bottom-up" e "top-down" para análise de programas com restrições, e proposta de um modelo para o projeto.
- Estudo do problema de tamanho de grão em programas com restrições, e desenvolvimento de técnicas de análise alternativas para o escalonamento de objetivos em sistemas distribuídos. Para tal será necessário generalizar as técnicas de análise para programas determinsem falhas para restrições, e ser capaz de estimar o custo da avaliação de restrições na análise de tamanho de grão.

Um das contribuições mais interessantes que esperamos obter nesta tarefa é a interação entre as técnicas de análise desenvolvidas para as linguagens de programação em lógica com restrições e as desenvolvidas para as linguagens funcionais. Assim como o desenvolvimento inicial da área para programação em lógica foi possível devido a esta interação, acreditamos que chegou a altura de considerar a investigação recente nas linguagens funcionais.

O desempenho do sistema paralelo dependerá fortemente do desempenho das ferramentas de compilação a desenvolver. O objetivo desta tarefa será o de implementar as seguintes ferramentas de compilação:

- Compilação de Prolog com restrições: o objetivo será o de simplificar as restrições fornecidas pelo usuário de forma a maximizar eficiência. Esta tarefa decorrerá fundamentalmente em NMSU.
- Detecção de independência entre objetivos: o objetivo será o de implementar um analisador global capaz de estabelecer em tempo de compilação as condições necessárias para o estabelecimento de independência entre objetivos. Nessa implementação tentaremos reaproveitar ferramentas já existentes para a geração de análise global, tais como CIAO [10].
- Análise de Granularidade: esta ferramenta tem como objetivo gerar informações sobre os tamanhos de objetivos e cláusulas no programa que possam ser utilizadas em tempo de execução para guiar estratégias de escalonamento. Estes tamanhos estão relacionados a tamanhos de tarefas intrinsicamente sequenciais e quantidades de paralelismo E e OU existente nas aplicações. Esta ferramenta poderá gerar dois tipos de informação: (i) simplificada, isto é, estimativas de tamanhos de tarefas diretamente utilizadas pelo escalonador ou (ii) complexas, onde as informações geradas consistem de expressões a serem computadas durante a execução.
- Detecção de dependência entre objetivos: o objetivo será estudar quais as condições para executar objetivos e restrições que compartilham variáveis, em paralelo.
- Geração de código especializado para o hardware FPGA: este módulo é o responsável pela geração das redes neuronais artificiais correspondentes aos conjuntos de cláusulas identificados como críticos, posterior geração dos circuitos digitais implementando as RNAs e, finalmente, seu mapeamento nos bancos de FPGAs.

4.1.3 Ferramentas de suporte

Nesta tarefa serão desenvolvidas ferramentas de suporte para a:

- Vizualização: será desenhada uma ferramenta de vizualização gráfica capaz de representar a execução E/OU. A ferramenta deverá ser fácilmente reconfigurável, de forma a permitir diferentes níveis de análise.
- Facilidades de Depuração: serão necessárias tanto para o desenvolvimento dos protótipos paralelos, como para o estudo das aplicações.

4.2 Desenvolvimento de aplicações

Nesta tarefa, pretendemos estudar aplicações pré-existentes e desenvolver novas aplicações em várias áreas de interesse: Bancos de Dados Dedutivos, mineração de dados, alocação de recursos, aplicações na WEB, programação em lógica indutiva, etc. Mais em detalhe, eis as principais aplicações a estudar:

- Através do projeto NSF correspondente, obteremos acesso às aplicações comerciais para restrições desenvolvidas pela companhia americana ALS, Inc.
- Através da nossa colaboração com NMSU, obteremos também acesso às aplicações de análise de linguagem natural desenvolvidas pelo Computing Research Laboratory de NMSU.
- Através da colaboração com Bell Labs, serão desenvolvidas aplicações de programação com restrições para pesquisa na WEB e para otimização de redes de chaveamento.
- Através da nossa colaboração com a Universidade do Porto, serão estudadas aplicações de Programação em Lógica Indutiva para o controle da decolagem e aterrisagem de aeronaves e para o desenho de fármacos.

4.3 Análise de Performance

O objetivo desta tarefa será o de validar o desempenho do sistema, através tanto de simulação como de experimentação em máquinas reais. Técnicas de simulação permitem uma análise controlada e detalhada do desempenho, que pode ser particularmente importante para a otimização do sistema [46, 47, 45, 53]. Os simuladores a usar são inicialmente MINT de Rochester e SimICS da SICS, e esperamos ainda avaliar o simulador SimOS de Stanford. De assinalar que simulação não pode substituir a experimentação com máquinas reais, para este último objetivo pretendemos usar as plataformas Beowulf a adquirir pelo projeto.

A análise de performance será efetuada com as aplicações obtidas na tarefa “Aplicações”. Para simplificar as simulações poderá ser usado inicialmente um conjunto de programas de teste simples, que permita estimar o desempenho do sistema.

Os principais parâmetros a estudar na tarefa são escalabilidade, quantidade de paralelismo existente nas aplicações, vantagens de utilização de diferentes técnicas para cada aplicação, tempo de desenvolvimento de aplicações utilizando as técnicas.

4.4 Avaliação do Projeto

O acompanhamento do projeto será realizado uma vez por ano. Pretendemos realizar workshops nacionais para avaliação interna. Além dos workshops, a forma de comunicação se dará principalmente através da Internet incluindo mail e página do projeto.

5 Estrutura de Colaboração entre os grupos

A colaboração e interação entre os grupos são mostrados na figura 5.

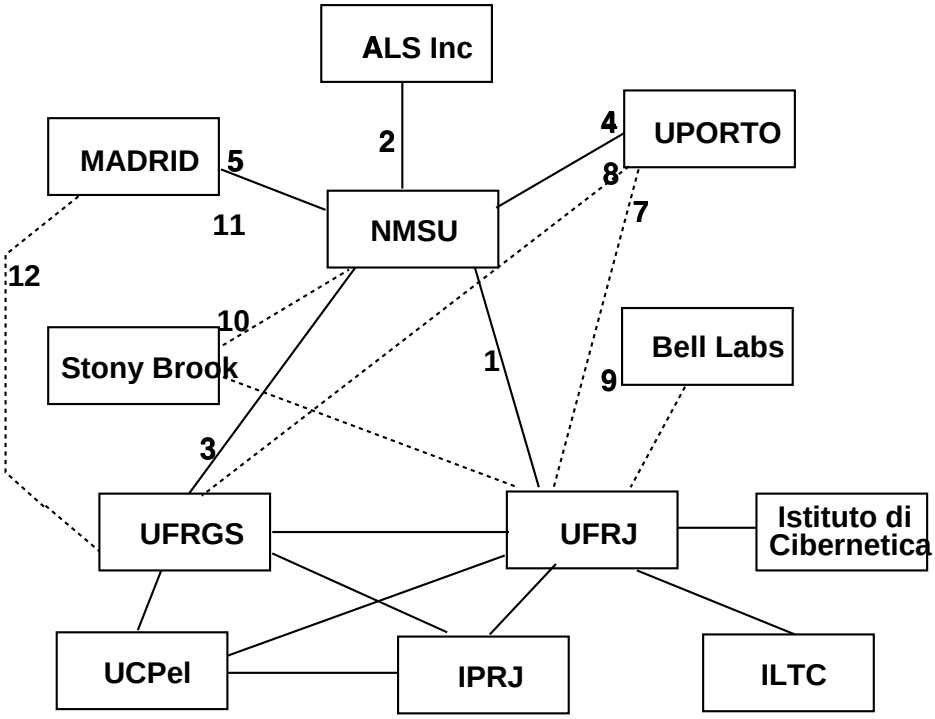

Figura 1: Estrutura de Colaboração (todas as instituições realizam a tarefa 4)

As ligações pontilhadas mostradas no grafo correspondem a cooperação do tipo consultoria. Ligações cheias correspondem a cooperação em tarefas específicas. Os números mostrados em cada ligação representam as tarefas em que os parceiros estarão envolvidos.

NMSU será o principal responsável pelo desenvolvimento da máquina abstrata paralela que incorpora paralelismo E e OU em SMPs, DMPs, e clusters de SMPs. NMSU também será responsável por desenvolver otimizações para a máquina paralela. A base do sistema em NMSU será a máquina abstrata fornecida por ALS, Inc, enquanto a UFRJ e UFRGS terão como base a máquina abstrata de domínio público do sistema clp(fd). Esta implementação estará baseada em técnicas de software DSM [30, 31].

UFRJ e UFRGS serão responsáveis principalmente pelo desenvolvimento das ferramentas de compilação (analisador de granularidade, analisador de dependências, compilação de restrições, compilação de partes críticas de coódigo fonte migradas FPGA) e dos algoritmos de escalonamento.

Pontelli, Gupta, Geyer e Santos Costa terão forte colaboração no desenvolvimento da máquina abstrata, compilador e analisadores estáticos. O trabalho de desenvolviemnto dos analisadores será feito em colaboração com o grupo de Manuel Hermenegildo em Madrid.

O grupo de Santos Costa, Dutra e Geyer desenvolverá os escalonadores que serão utilizados por Gupta e Pontelli. O trabalho em DMP será também em colaboração com Fernando Silva na Universidade do Porto, dada sua grande experiência na área de implementação de sistemas Prolog distribuídos.

UFRJ também conduzirá detalhada análise de performance dos sistemas desenvolvidos em colaboração com NMSU. Esta tarefa será liderada por Santos Costa.

Aplicações reais serão desenvolvidas por Juliana Freire de Bell Labs e pelo grupo na UFRJ. Utilizaremos também aplicações de clientes de ALS, Inc. através de um acordo entre NMSU e

ALS, Inc.

A parte do projeto relativa á migração de partes críticas de código fonte para hardware será feita exclusivamente pelos grupos da UFRJ, ILTC, UFRGS e Istituto di Cibernetica com a assistência de NMSU. Priscila Lima, Felipe França, Inês Dutra e Massimo De Gregorio serão os responsáveis por esta tarefa.

Os grupos do IPRJ, UFRGS, UCPel e UFRJ serão responsáveis principalmente pela interação na parte de análise global dos programas para geração de informação relativa a paraleização de programas e análise de granularidade.

Abaixo segue o cronograma do projeto.

Tarefas	Ano 1		Ano 2		Ano 3	
	1	2	1	2	1	2
1.a.i	√	√	√	√	√	
1.a.ii	√	√	√	√	√	
1.a.iii		√	√	√	√	
1.a.iv		√	√	√	√	
1.b.i	√	√				
1.b.ii	√	√	√			
1.b.iii	√	√	√	√		
1.b.iv					√	√
1.b.v	√	√	√	√		
1.c	√	√	√	√		
2.a					√	√
2.b		√	√	√		
2.c		√	√	√		
3		√	√	√	√	√
4		√		√		√

6 Recursos Necessários

6.1 Custeio

Estão previstas atividades de ensino e pesquisa relacionadas com o projeto, assim como participação em eventos nacionais e internacionais, organização de evento nacional no tema do projeto com possibilidade de convite a pesquisadores com produção científica relevante a nível internacional.

O projeto contará com a participação ativa de alunos de iniciação científica, alunos de mestrado e doutorado.

Valor destinado a bolsas (em Reais):

- Bolsa de longa duração
  - Uma (1) bolsa DTI-E por 24 meses: R$ 30.000,00
    Tarefas associadas: 1.a e 1.b.
- Bolsas de curta duração
  - 6 estágios no exterior com duração de um mês: R$ 30.000,00
    Tarefas associadas: 1.a (2 estágios) e 1.b (2 estágios).
  - 9 participações em eventos internacionais com duração de uma semana: R$ 22.500,00
  - 4 bolsas AEV com duração de um mês: R$ 16.000,00
    Tarefas associadas: 1.a.i, 1.a.ii, 1.b.ii, 2.a e 2.b.

– 24 participações em eventos nacionais com duração de uma semana: R$ 19.200,00

Subtotal: R$ 117.700,00

6.2 Capital

- Equipamentos
  - Oito (8) estações de trabalho dual Pentium: R$ 30.000,00.
    Tarefas associadas: 1, 2 e 3.
  - Uma (1) placa PCI contendo um banco de FPGAs e sistema de desenvolvimento: R$ 6.000,00.
    Tarefas associadas: 1 e 2.
- Bibliografia: R$ 2.000,00.

Subtotal: R$ 38.000,00

6.3 Total Geral

Custeio:	R$ 117.700,00
Capital:	R$ 38.000,00
Total:	R$ 155.700,00

7 Relevância do Projeto

Programação lógica com restrições tem se tornado cada dia mais popular e atualmente há pelo menos uma conferência internacional sobre o tema. O tema "solução de problemas com restrições" (constraint problem solving) recentemente ganhou espaço na forma de seção inteira em uma das conferências mais importantes na área de Inteligência Artificial (IJCAI – International Joint Conference on Artificial Intelligence).

Sistemas de programação lógica com restrições têm aplicações em vários problemas difíceis cuja complexidade de tempo é exponencial. Estes sistemas permitem a implementação de aplicações reais em várias áreas do conhecimento que se adequam ao paradigma de programação lógica: Processamento de Linguagem Natural, Otimização, Alocação de Recursos, Bancos de Dados, etc. Os mecanismos de poda utilizados pelos "solvers" das equações/inequações (revised simplex, forward checking, lookahead, arc consistency etc) permitem que problemas com entradas suficientemente grandes possam rodar em tempo tolerável.

Além desta vantagem, programação lógica permite que o programador se concentre na solução do problema sem precisar se preocupar com questões de controle ou características específicas da plataforma de hardware.

Vários grupos de pesquisa na Europa, Japão e EUA (p. e., University of Bristol, Universidad Politecnica de Madrid, Universidade Nova de Lisboa, Universidade do Porto, SUNY at Stony Brook, NMSU, SICS IC-PARC, ICOT), assim como empresas e órgãos governamentais, financiam e fazem pesquisa na área de Programação Lógica por acreditarem no seu potencial. Empresas como a British Telecom (Inglaterra), Ericsson (Suécia) e outras, investem grande parte de seu tempo e verbas em pesquisa na área de programação lógica, principalmente em desenvolvimento de aplicações. Recentemente na última Joint International Conference and Symposium on Logic Programming (Manchester, Junho 98), houve uma sessão sobre aplicações industriais de programação lógica com restrições, onde vários palestrantes mostraram que programar em PLC pode ser muito mais rentável que programar em uma linguagem convencional [12]. Uma das palestras mostrou uma comparação de performance entre um sistema conhecido para verificação de

modelos implementado em C (SPIN) e um sistema de verificação de modelos (XMC) idêntico escrito em XSB [44] (Prolog com tabulação e restrições) [41]. Mostrou-se que a performance do XMC era muito melhor do que a do SPIN para resolver o mesmo problema. Além disso o tempo de desenvolvimento da aplicação foi muito menor no caso do XMC do que no caso do SPIN.

Programas escritos em linguagens lógicas são fáceis de ler, escrever e manter. Restrições adicionam uma vantagem à linguagem através de um mecanismo mais poderoso de unificação. Paralelismo adiciona ganho de performance, facilidade de programação e portabilidade já que a exploração de paralelismo é feita de forma implícita.

Acreditamos que a implementação de um sistema deste porte seria de extrema utilidade em vários aspectos acadêmicos e cooperação com programadores de aplicações. O projeto propõe as seguintes contribuições:

- Ensino e pesquisa: alunos teriam a oportunidade de conhecer em detalhes, com estudo teórico e prático, pelo menos quatro áreas em Ciência da Computação: Inteligência Artificial, Programação Lógica, Paralelismo e Programação com Restrições. O envolvimento de alunos no projeto é de extrema importância para uma formação acadêmica sólida.
- Desenvolvimento de uma ferramenta de alto nível, robusta, para desenvolvimento de aplicações em várias áreas.
- Colaboração a nível nacional e internacional com outros grupos de pesquisa em implementações de sistemas de programação lógica.
- Solução de problemas importantes tais como alocação de recursos, mineração de dados e escalonamento com o desenvolvimento de aplicações no contexto do projeto.

Referências

[1] IEEE Standart VHDL Language Reference Manual. Technical report, The Institute of Electrical and Electronic Engineers, 1988. IEEE Standard 1076-1987.

[2] A. Aiello, E. Burattini, and G. Tamburrini. Purely neural, rule-based diagnostic systems. *International Journal of Intelligent Systems*, (10):735–749, 1995. Part I. Production rule.

[3] A. Aiello, E. Burattini, and G. Tamburrini. Neural nets and rule-based reasoning. *Fuzzy Logic and Expert Systems Applications*, 1997. Part I. Production rule.

[4] C. L. Amorim, R. Bianchini, G. Silva, R. Pinto, M. Hor-Meyll, M. de Maria, L. Whately, and J. Barros Jr. A Segunda Geração de Computadores de Alto Desempenho da COPPE/UFRJ. In *VII Simpósio Brasileiro de Arquitetura de Computadores, SBAC-PAD*, July 1996.

[5] Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team. A case for networks of workstations: Now. *IEEE Micro*, February 1995.

[6] J. L. V. Barbosa and C. F. R. Geyer. Análise de Complexidade na Programação em Lógica: Taxonomia, Modelo Granlog e Análise OU. In *XXIII Conferência Latino Americana de Informática, V Encontro Chileno de Computação da Sociedade Chilena de Computação (CLEI – PANEL'97)*, November 1997.

[7] Jorge Luis Victória Barbosa. GRANLOG: Um Modelo para Análise Automática de Granulosidade na Programação em Lógica. Master's thesis, Universidade Federal do Rio Grande do Sul, Instituto de Informática, Curso de Pós-Graduação em Ciência da Computação, 1996.

[8] Anthony Beaumont and David H. D. Warren. Scheduling Speculative Work in Or-Parallel Prolog Systems. In *Proceedings of the Tenth International Conference on Logic Programming*, pages 135–149. MIT Press, June 1993.

[9] R. Bianchini, L. Kontothanassis, R. Pinto, M. De Maria, and M. Abud C. L. Amorim. Hiding Communication Latency and Coherence Overhead in Software DSMs. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS7)*, October 1996. Also available as Technical Report ES-356/95, COPPE/Systems Engineering, Federal University of Rio de Janeiro.

[10] Francisco Bueno, M. García de la Banda, and M. V. Hermenegildo. Effectiveness of Abstract Interpretation in Automatic Parallelization: A Case Study in Logic Programming. *ACM TOPLAS*, 1998.

[11] Ernesto Burattini and Massimo De Gregorio. A Hybrid Neurosymbolic Model for Intelligent Traffic Control Systems. In *Anais do Workshop em Inteligência Computacional: Projetos ICOM e IPAC, Protem III-CC, CNPq*, pages 48–49, Setembro 1997. Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ, ES-450/97.

[12] Vanusa Menditi Calegario and Inês de Castro Dutra. Performance Comparison between Conventional and Logic Programming Systems. Technical Report ES-478/98, COPPE/Systems Engineering and Computer Science, Setembro 1998.

[13] Luís Fernando P. Castro, Vítor Santos Costa, Manuel E. Correia, and Fernando Silva. DAOS – Distributed And-Or Scheme. Research Report, Instituto de Informática, UFRGS, 1998.

[14] A. P. Centeno and C. Geyer. Penelope – Um Modelo de Escalonador Hierárquico para o Sistemas Plosys. In *X Simpósio Brasileiro de Arquitetura de Computadores, SBAC-PAD*, Setembro 1998.

[15] Philippe Codognet and Daniel Diaz. `wamcc`: Compiling Prolog to C. In *12th International Conference on Logic Programming*. The MIT Press, 1995.

[16] Manuel Eduardo Correia, Vítor Santos Costa, and Fernando Silva. The SBA: Exploiting Orthogonality in AND–OR Parallel Systems. In *ILPS'97, Port Jefferson, NY, USA,*, 1997.

[17] P. Cousot and R. Cousot. Abstract Interpretation; A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conf. Rec. 4th ACM Symp. on Prin. of Programming Languages*, pages 238–252, 1977.

[18] Cristiano A. da Costa and C. Geyer. Uma Proposta de Escalonamento Distribuído para Exploração do Paralelismo na Programação em Lógica. In *X Simpósio Brasileiro de Arquitetura de Computadores, SBAC-PAD*, Setembro 1998.

[19] Daniel Diaz and Philippe Codognet. A Minimum Extension of the WAM for clp(FD). In *10th International Conference on Logic Programming*, pages 774–792. The MIT Press, 1993.

[20] I. C. Dutra. Strategies for Scheduling And- and Or-Work in Parallel Logic Programming Systems. In *Proceedings of the 1994 International Logic Programming Symposium*, pages 289–304. MIT Press, 1994. Also available as technical report CSTR-94-09, from the Department of Computer Science, University of Bristol, England.

[21] I. C. Dutra. Distributing And-Work and Or-Work in Parallel Logic Programming Systems. In *Proceedings of the 29th Hawaii International Conference on System Sciences*, pages 646–655. IEEE, 1996.

[22] Victor M. G. Ferreira. *Mapeando Redes Neurais Artificiais para Processamento de Inferências em FPGAs*. UFRJ, Escola de Engenharia, Dez 1997.

[23] V.M.G. Ferreira and F.M.G. Fran¸ca. Weightless circuit synthesis of weighted ANNs. In *Anais do Workshop em Inteligência Computacional: Projetos ICOM e IPAC, Protem III-CC, CNPq*, pages 50–52, Setembro 1997. Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ, ES-450/97.

[24] V.M.G. Ferreira and F.M.G. Fran¸ca. Weightless implementations of weighted neural networks. In *IV Brazilian Symposium on Neuronal Networks*, pages 53–54, Dec 1997.

[25] Pascal Fradet and Daniel le Métayer. Structured Gamma. Technical Report IRISA-989, IRISA, Mar 1996.

[26] T. Frühwirth, A. Herold, V. Küchenhoff, T. Le Provost, P. Lim, E. Monfroy, and M. Wallace. Constraint Logic programming – An Informal Introduction. Technical Report ECRC-93-5, European Computer-Industry Research Centre, Feb 1993.

[27] S. Gregory and R. Yang. Parallel Constraint Solving in Andorra-I. In *Proceedings of the 1992 International Conference on Fifth Generation Computer Systems*, pages 843–850. ICOT, 1992.

[28] G. Gupta and Enrico Pontelli. Stack-Splitting Based Or-Parallelism. NMSU Technical Report, New Mexico State University, 1998.

[29] Gopal Gupta, Enrico Pontelli, and Manuel Hermenegildo. &ACE: A High Performance Parallel Prolog System. In *Proceedings of the First International Symposium on Parallel Symbolic Computation, PASCO'94*, 1994.

[30] P. Keleher, A. L. Cox, and W. Zwaenepoel. Lazy Release Consistency for Software Distributed Shared Memory. In *19rd Annual International Conference on Computer Architecture*, pages 13–21, May 1992.

[31] P. Keleher, S. Dwarkadas, A. Cox, and W. Zwaenepoel. TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems. In *USENIX Winter'94 Technical Conference*, pages 17–21, Jan 1994.

[32] P. M. V. Lima. Resolution-Based Inference with Artificial Neural Networks: Avoiding Cyclic Dependencies. In *Proceedings of the 2nd Brazilian Conference on Neural Networks (CBRN-95), Curitiba, Brazil*, 1995.

[33] P. M. V. Lima. Towards Neural Computation of REFORMed Programs. In *Proceedings of the Workshop on High Performance Logic Programming Systems, European Summer School on Logic, Languages and Implementation (ESSLLI'96), Prague*, August 1996.

[34] P. M. V. Lima. *Resolution based inference with Artificial Neural Networks*. PhD thesis, Imperial College London, 1998. in preparation.

[35] Nai-Wei Lin and Saumya K. Debray. Cost Analysis of Logic Programs. Technical Report, Department of Computer Science, The University of Arizona, August 1992.

[36] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, second edition, 1987.

[37] Kim Marriot and Peter J. Stuckey. *Programming with Constraints*. MIT Press, 1998.

[38] C. S. Mellish. Some Global Optimizations for a Prolog Compiler. *The Journal of Logic Programming*, 2(1), April 1985.

[39] E. Morel, J. Briat, J. Chassin de Kergommeaux, and C. Geyer. Side-Effects in PloSys OR-parallel Prolog on Distributed Memory Machines. In *JICSLP'96 Post-Conference Workshop on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages, Bonn, Germany*, September 1996.

[40] A. Mycroft. *Abstract Interpretation and Optimising Transformations for Applicative Programs*. PhD thesis, University of Edinburgh, 1981.

[41] Y. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, S. Smolka, T. Swift, and D. S. Warren. Efficient Model Checking using tabled resolution. In *CAV'97*, 1997.

[42] Daniel Ridge, Donald Becker, Phillip Merkey, Thomas Sterling Becker, and Phillip Merkey. Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs. In *IEEE Aerospace 1997*, 1997.

[43] Ricardo Rocha, Fernando Silva, and V. Santos Costa. On Applying Or-Parallelism to Tabled Evaluations. In *Post-ICLP'97 Workshop on Tabling in Logic Programming*, July 1997. Leuven, Belgium, also published as Technical Report DCC-97-2, DCC - FC & LIACC, Universidade do Porto, April, 1997.

[44] K. Sagonas, T. Swift, D.S. Warren, J. Freire, and P. Rao. The XSB programmer's manual. Technical report, 1997. Available at http://www.cs.sunysb.edu/~sbprolog.

[45] V. Santos Costa and R. Bianchini. Optimising Parallel Logic Programming Systems for Scalable Machines. In *Proceedings of the EUROPAR'98*, pages 831–841, Sep 1998.

[46] V. Santos Costa, R. Bianchini, and I. C. Dutra. Evaluating the Impact of Coherence Protocols on Parallel Logic Programming Systems. In *Proceedings of the 5th EUROMICRO Workshop on Parallel and Distributed Processing*, pages 376–381, 1997. Also available as technical report ES-389/96, COPPE/Systems Engineering, May, 1996.

[47] V. Santos Costa, R. Bianchini, and I. C. Dutra. Parallel Logic Programming Systems on Scalable Multiprocessors. In *Proceedings of the 2nd International Symposium on Parallel Symbolic Computation, PASCO'97 [**?**]*, pages 58–67, July 1997.

[48] V. Santos Costa, D. H. D. Warren, and R. Yang. Andorra-I: A Parallel Prolog System that Transparently Exploits both And- and Or-Parallelism. In *Third ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, pages 83–93. ACM press, April 1991. SIGPLAN Notices vol 26(7), July 1991.

[49] Ví tor Santos Costa. Cowl: Prolog and the magic cow. Technical note, LIACC, Universidade do Porto, 1998.

[50] Vítor Santos Costa, David H. D. Warren, and Rong Yang. The Andorra-I Preprocessor: Supporting full Prolog on the Basic Andorra model. In *Proceedings of the Eighth International Conference on Logic Programming*, pages 443–456. MIT Press, 1991.

[51] Kish Shen, Vítor Santos Costa, and Andy King. A New Metric for Controlling Granularity for Parallel Execution. In *Joint International Conference and Symposium on Logic Programming*, Manchester, UK, June 1998.

[52] Fernando M. A. Silva. *Implementations of Logic Programming Systems*, chapter Or-Parallelism on Distributed Shared Memory Architectures. Kluwer Academic Pub., 1994.

[53] Márcio G. Silva, Inês C. Dutra, Ricardo Bianchini, and Vítor Santos Costa. The Influence of Computer Architectural Parameters on Parallel Logic Programming Systems. In Gopal Gupta, editor, *Workshop on Practical Aspects of Declarative Languages (PADL99), Lecture Notes in Computer Science 1551*, pages 122–136, January 1999. Also available as Technical Report ES/477-98, COPPE Systems Engineering, Sep/98.

[54] Synopsys. *VHDL Compiler Reference Version 3.3a*. Synopsys, Inc., Mountain View, CA, Apr 1995.

[55] Pascal van Hentenryck. *Constraint Satisfaction in Logic programming*. MIT Press, 1989.

[56] P. K. Vargas, C. Geyer, and I. C. Dutra. TAMAGOSHI – Plataforma para Avaliação de Escalonamento de Tarefas em Programação em Lógica Paralela. In *X Simpósio Brasileiro de Arquitetura de Computadores, SBAC-PAD*, Setembro 1998.

[57] Rong Yang, Vítor Santos Costa, and David H. D. Warren. The Andorra-I Engine: A parallel implementation of the Basic Andorra model. In *Proceedings of the Eighth International Conference on Logic Programming*, pages 825–839. MIT Press, 1991.

[58] X. Zhong, E. Tick, S. Duvvuru, L. Hansen, A. V. S. Sastry, and R. Sundararajan. Towards an Efficient Compile-Time Granularity Analysis Algorithm. Technical Report CIS-TR-91-19, University of Oregon, Department of Computer and Information Science, 1991.