Uma Plataforma de Baixo Consumo Energético, Flexível e Barata para a Internet das Coisas

PROPONENTE:
Francisco Figueiredo Goytacaz Sant'Anna
francisco@ime.uerj.br

INSTITUIÇÃO DE EXECUÇÃO:
Departamento de Informática e Ciências da Computação
Instituto de Matemática e Estatística (IME)
Universidade Estadual do Rio de Janeiro (UERJ)

Chamada Universal MCTIC/CNPq 2018

17 de agosto de 2018

1 Project

1.1 Project title (in portuguese):

Uma Plataforma de Baixo Consumo Energético, Flexível e Barata para a Internet das Coisas

1.2 Project title (in english):

A Low-Power, Flexible, and Cheap Platform for the Internet of Things

1.3 Keywords (in portuguese):

(Enter keywords separated by comma, 1 word minimum, 6 words maximum)
iot, arduino, eficiência energética, linguagem síncrona

1.4 Keywords (in english):

(Enter keywords separated by comma, 1 word minimum, 6 words maximum)
iot, arduino, low power, synchronous language

1.5 Objetivo (Em Português):

Criar uma plataforma de hardware e software para pesquisa e desenvolvimento de aplicações para a Internet das Coisas com as seguintes características:

- Baixo Consumo Energético: O hardware deve possuir modos de economia de energia para todos os seus componentes, sejam eles o microcontrolador, sensores, ou módulos de rádio. O software será baseado em uma linguagem de programação ciente de energia, que seja capaz de detectar quando os componentes estão ociosos para colocá-los em modo de espera (standby mode) automaticamente, sem a intervenção do programador.

- Flexível: O hardware deve prever conexões para uma variedade de sensores e transceptores de rádio frequência. Em particular, os módulos de rádio mais populares devem ser todos acopláveis externamente ao hardware. O software deve ser modular, de maneira que somente os

drivers dos dispositivos de interesse sejam compilados junto com as aplicações.

- Barato: O hardware deve usar microcontroladores e módulos "de platereira"que sejam encontrados com facilidade no mercado brasileiro para compras em pequenas quantidades e a custo baixo. O plataforma de software deve ser toda baseada em software livre.

Como principais desafios, o hardware deve possuir mecanismos flexíveis que permitam desabilitar periféricos via software e, principalmente, a linguagem de programação deve oferecer mecanismos automáticos para colocar o hardware em modo de espera.

1.6 Objetivo (Em Inglês):

2 Abstract

2.1 Abstract (in portuguese):

De acordo com a Agência Internacional de Energia (AIE) [1], o número de dispositivos conectados deve atingir 50 bilhões até 2020 com a expansão da Internet das Coisas (IoT). A maior parte do consumo de energia nesses dispositivos será em *modo de espera* (aka *standby mode*), quando eles não estão transmitindo ou processando dados. Em particular, o modo de espera é responsável por aproximadamente 10–15% do consumo residencial. Também estima-se que as emissões de CO_2 relacionadas ao modo de espera mundialmente é equivalente ao de 1 milhões de carros.

Os efeitos substanciais do consumo em modo de espera, aliados ao crescimento estimado da IoT, tornou o modo de espera para dispositivos conectados um dos seis pilares do *Plano de Ação para Eficiência Energética do G20*[1]. No entanto, o uso efetivo do modo de espera requer grandes esforços de software e hardware para detectar períodos de inatividade nos dispositivos, identificar periféricos que devem permanecer ligados, e aplicar os modo mais econômico sempre que possível.

Sendo assim, este projeto tem como principal objetivo desenvolver uma plataforma de hardware e software de baixo consumo energético para pesquisa e desenvolvimento de aplicações para a Internet das Coisas.

[1] G20's Energy Efficiency Action Plan: `https://www.iea-4e.org/projects/g20`

que faça uso do modo de espera de maneira eficiente.
com as seguintes características:
Nosso trabalho com Céu e suporte automático

2.2 Abstract (in english):

3 Scientific Knowledge field:

- Predominant area:
  - Sistemas de Computação
- Related Areas:
  - Sistemas de Telecomunicações
  - Linguagens de Programação

a) Identificação do projeto, incluindo título, palavras-chave e resumo; b) Dados do proponente e equipe; c) Área(s) do conhecimento predominante(s); d) Instituição(ões) participante(s); e) Objetivos geral e específicos; f) Metodologia proposta; g) Etapas de execução do projeto com respectivo cronograma de atividades; h) Produtos esperados como resultado da execução do projeto, com previsão de cronograma de entregas anuais; i) Potencial de impacto dos resultados do ponto de vista técnico-científico, de inovação, difusão, sócio-econômico e ambiental; j) Colaborações ou parcerias já estabelecidas para a execução do projeto; k) Perspectivas de colaborações interinstitucionais para a execução do projeto; l) Recursos financeiros de outras fontes aprovados para aplicação no projeto; m) Disponibilidade efetiva de infraestrutura e de apoio técnico para o desenvolvimento do projeto; n) Orçamento detalhado.

- Até 36 meses

Identificação da proposta; Qualificação do principal problema a ser abordado; Objetivos e metas a serem alcançados; Indicadores de acompanhamento; Metodologia a ser empregada; Principais contribuições científicas, tecnológicas ou de inovação da proposta; Orçamento detalhado; Cronograma de atividades; Identificação de todos os participantes do projeto: Grau de interesse e comprometimento de empresas com o escopo da proposta, quando for o caso; Indicação de colaborações ou parcerias já estabelecidas com outros centros de pesquisa na área; Disponibilidade efetiva de infraestrutura e

de apoio técnico para o desenvolvimento do projeto; Estimativa dos recursos financeiros de outras fontes que serão aportados pelos eventuais Agentes Públicos e Privados parceiros e No caso das solicitações de bolsas de IC e/ou AT deverá ser apresentado, juntamente com o projeto de pesquisa, um plano das atividades a serem desenvolvidas pelo bolsista. Não é necessário indicar o nome do candidato no momento da submissão da proposta, apenas descrever o perfil desejado para o futuro bolsista.

4 Identificação da Proposta

- **Título:** Recuperação Automática de Erros para Parsers Baseados em Gramáticas de Expressões de Parsing;
- **Palavras-chave:** gramáticas de expressões de parsing (PEGs), parsing, recuperação de erros
- **Áreas do Conhecimento:** Ciência da Computação — Linguagens Formais e Autômatos / Linguagens de Programação;
- **Instituição:** Escola de Ciências e Tecnologia (ECT) / UFRN;

5 Equipe

- Sérgio Queiroz de Medeiros
  - Função: Pesquisador Proponente e Coordenador;
  - Lattes: http://lattes.cnpq.br/0310395336626784;
  - Instituição: Universidade Federal do Rio Grande do Norte
- Fabio Mascarenhas de Queiroz
  - Função: Professor Colaborador;
  - Lattes: http://lattes.cnpq.br/2273723591083358;
  - Instituição: Universidade Federal do Rio de Janeiro
- Gilney de Azevedo Alves Junior
  - Função: Colaborador (Aluno de Graduação);

- Lattes: http://lattes.cnpq.br/8502686732277287;
- Instituição: Universidade Federal do Rio Grande do Norte

Também podem fazer parte da equipe do projeto estudantes de mestrado vinculados ao Programa de Pós-Graduação em Engenharia Software (PPGSW) da UFRN, do qual o proponente do projeto é membro permamente.

6 Introdução

Ambientes de Desenvolvimento Integrado (*Integrated Development Environments* - IDEs) são uma ferramenta essencial no processo de desenvolvimento de software atual. Para que uma IDE possa oferecer funcionalidades como refatoração automática e *coding complete*, o parser associado a ela deve construir uma Árvore de Sintaxe Abstrata (*Abstract Syntax Tree* - AST) mesmo para programas sintaticamente inválidos. Tal parser deve ter portanto a capacidade de se recuperar de erros sintáticos e continuar processando um programa sintaticamente inválido, de modo a produzir uma AST válida e que captura uma boa quantidade de informação desse programa.

Gramáticas de Expressões de Parsing (*Parsing Expression Grammars* - PEGs) [**?**, **?**] são um formalismo que permite a geração de parsers descendentes (*top-down*) recursivos a partir de uma descrição formal que é visualmente semelhante à descrição de uma Gramática Livre de Contexto (GLC) [**?**]. PEGs possuem um operador de escolha ordenada que não permite expressar ambiguidade, o que é desejável quando descrevemos uma linguagem para uma máquina (e.g., a sintaxe de uma linguagem de programação). Devido a esse operador de escolha ordenada, o parser gerado a partir da descrição de uma PEG possui uma forma restrita de retrocesso (*backtracking*).

Falhas rotuladas são uma extensão conservativa de PEGs que permite adicionar um mecanismo de recuperação de erros para parsers baseados em PEGs [**?**]. Cada falha rotulada pode ser associada a uma expressão de recuperação, que deve consumir a entrada após uma falha até que seja possível reconhecer uma construção sintaticamente válida do programa.

O uso de falhas rotuladas exige que a gramática seja anotada para indicar quando uma determinada falha deve ser lançada. Realizar manualmente essa anotação da gramática demanda tempo do desenvolvedor, e pode ser uma tarefa difícil para desenvolvedores menos experientes.

Após anotar a gramática com falhas rotuladas, ainda é necessário associar uma expressão de recuperação a cada falha. Uma boa expressão de recuperação deve ser capaz de sincronizar o parser com a entrada após um erro sem descartar muita informação da entrada. Dessa forma, é possível gerar uma AST rica, que captura a maior parte da informação, para programas sintaticamente inválidos.

Em virtude dessas dificuldades, muitos parsers baseados em PEGs não possuem um mecanismo de recuperação de erros, ou não possuem um mecanismo de recuperação robusto (i.e., que produz uma AST rica), o que torna inviável o uso desses parsers em IDEs.

7 Objetivos Geral e Específicos

Levando em consideração os problemas descritos anteriormente, nossos objetivos gerais são:

1. projetar e implementar um algoritmo para anotar automaticamente uma gramática PEG com falhas rotuladas;

2. projetar e implementar uma abordagem para associar automaticamente expressões de recuperação a falhas rotuladas, de modo que um parser baseado em PEGs produza ASTs ricas para entradas sintaticamente inválidas.

Para alcançar o primeiro objetivo, teríamos que cumprir os seguintes objetivos específicos:

1. Propor um algoritmo para inserir falhas rotuladas automaticamente em uma PEG;

2. Implementar o algoritmo proposto;

3. Aplicar o algoritmo a PEGs que descrevem linguagens de programação e avaliar a saída dos parsers resultantes para entradas sintaticamente válidas e inválidas.

Para alcançar o segundo objetivo geral, teríamos que cumprir os seguintes objetivos específicos:

1. Propor uma abordagem para associar automaticamente expressões de recuperação a falhas rotuladas;

2. Implementar a abordagem proposta;

3. Aplicar a abordagem para diferentes parsers baseados em PEGs e avaliar a qualidade da AST gerada para programas sintaticamente inválidos.

8 Metodologia

Para alcançar o primeiro objetivo geral, devemos tomar como base da nossa pesquisa o algoritmo apresentado em [?], que propõe uma abordagem semiautomática para inserir falhas rotuladas em uma PEG. Pretendemos solucionar o problema da inserção indevida de falhas rotuladas, que leva o parser resultante a rejeitar entradas sintaticamente válidas.

Após definirmos um algoritmo que adiciona corretamente falhas rotuladas a uma gramática, iremos implementá-lo em um gerador de parsers baseado em PEGs. Com isso, vamos obter uma ferramenta que aceita como entrada a descrição de uma PEG, sem falhas rotuladas, e produz como resultado uma PEG anotada, com falhas rotuladas.

Iremos comparar então as PEGs anotadas automaticamente através dessa ferramenta com PEG anotadas manualmente e analisar o impacto das possíveis diferenças entre as gramáticas. Pretendemos inicialmente refazer a análise feita em [?] para a linguagem de programação Titan e em seguida realizar essa comparação com os parsers de outras linguagens de programação, como C e Java. Com base nos resultados obtidos, iremos ajustar o algoritmo e refazer os experimentos.

Para alcançar o segundo objetivo geral, iremos tomar como base estratégias de recuperação de erros sintáticos usadas por geradores de parsers populares como ANTLR [?, ?] e Coco/R [?], bem como estratégias descritas na literatura [?, ?, ?].

Após definir uma estratégia de recuperação de erros automática adequada para PEGs com falhas rotuladas, implementaremos uma ferramenta que usa essa estratégia. Tal ferramenta deve receber como entrada a descrição de uma PEG com falhas rotuladas e produzir como saída uma PEG onde cada falha rotulada está associada a uma expressão de recuperação.

Devemos notar que a entrada desta segunda ferramenta é a saída da primeira ferramenta, relacionada ao primeiro objetivo geral. Assim, geraremos parsers com recuperação de erros para as linguagens de programação que

utilizarmos nos experimentos associados ao primeiro objetivo geral.

Após obtermos parsers baseados em PEGs com recuperação de erros, iremos avaliar a qualidade das ASTs geradas por esses parsers para programas sintaticamente inválidos. Essa avaliação da qualidade da recuperação de erros se baseará na metodologia usada em trabalhos relacionados, tais como [?] e [?].

Devemos também comparar a AST gerada pelas ferramentas que desenvolveremos com as ASTs geradas por outros geradores de parsers, tais como ANTLR.

9 Cronograma de Atividades

A Tabela ?? apresenta um cronograma semestral com as principais atividades do projeto. Usamos nessa tabela a seguinte legenda para as atividades:

- A1: Projetar algoritmo para inserir falhas rotuladas em uma PEG;
- A2: Implementar o algoritmo da atividade A1;
- A3: Desenvolver parsers baseados em PEGs para linguagens de programação;
- A4: Aplicar o algoritmo da atividade A1 para parsers de linguagens de programação e avaliar o resultado;
- A5: Aprimorar a inserção de falhas automáticas e refazer a avaliação A4;
- A6: Definir uma abordagem para associar automaticamente expressões de recuperação a falhas rotuladas;
- A7: Implementar a abordagem definida em A6;
- A8: Aplicar a abordagem da atividade A6 para parsers de linguagens de programação e avaliar o resultado;
- A9: Refinar a abordagem da atividade A6 e refazer a avaliação A8.
- A10: Elaboração de artigos científicos.

10 Resultados Esperados

Ao final do primeiro ano do projeto os resultados esperados são:

1. Um algoritmo que anote um gramática PEG com falhas rotuladas;
2. Incorporação do algoritmo anterior a um gerador de parsers baseado em PEGs e disponibilização do mesmo para a comunidade;
3. Um artigo científico com os resultados parciais da pesquisa.

Ao final do segundo ano, o projeto deverá produzir os seguintes resultados:

1. Uma abordagem para associar automaticamente expressões de recuperação a falhas rotuladas;
2. Uma ferramenta, disponível para a comunidade, que a partir da descrição formal de uma PEG produz um parser com um mecanismo de recuperação de erros sintáticos razoável;
3. Um artigo científico com os resultados parciais da pesquisa.

Ao final do terceiro ano, devemos ter os seguintes resultados:

1. Uma ferramenta, disponível para a comunidade, que a partir da descrição formal de uma PEG produz um parser com um mecanismo de recuperação de erros sintáticos robusto;
2. Facilidade de usar parsers baseados em PEGs em IDEs;
3. Submissão de um artigo para um periódico científico.

11 Contribuições

A principal contribuição técnico-científica do trabalho será a construção de uma ferramenta que permitirá obter automaticamente, a partir de uma gramática PEG, um parser com um mecanismo robusto de recuperação de erros. Essa contribuição tornará viável o uso de parsers baseados em PEGs em IDEs. Além dessa contribuição principal, há outras que discutimos a seguir.

O algoritmo de inserção automática de falhas rotuladas, além de ser útil para a recuperação automática de erros, vai facilitar a tarefa de reportar os

erros sintáticos para um usuário. Após uma gramática PEG ser anotada com falhas rotuladas, para que o parser correspondente produza boas mensagens de erro será necessário apenas associar uma mensagem de erro a cada falha rotulada.

Uma outra contribuição da pesquisa é a aplicação dos resultados teóricos, através do desenvolvimento de ferramentas que tornem fácil a obtenção de parsers PEGs com recuperação automática de erros. Além dessas ferramentas, a pesquisa deve produzir parsers robustos para as linguagens de programação que serão utilizadas nos estudos de caso.

Os softwares relacionados à pesquisa possuem potencial de inovação na área de parsing/compiladores e podem vir a ser registrados.

Por fim, os artigos científicos associados à pesquisa devem estimular o desenvolvimento de outros geradores de parsers baseados em PEGs que suportem a recuperação automática de erros sintáticos.

12 Histórico de Pesquisa

O proponente deste projeto e o pesquisador Fabio Mascarenhas, membro da equipe do projeto, atuam na pesquisa relacionada a PEGs desde 2008. Inicialmente, estudaram a relação entre PEGs e outros formalismos usados para descrever linguagens. Entre os artigos relacionados a essa pesquisa destacamos *From regexes to parsing expression grammars* [**?**], que discute a correspondência entre expressões regulares, incluindo extensões comumente usadas por ferramentas de casamento de padrões, e PEGs; e o artigo *On the relation between context-free grammars and parsing expression grammars* [**?**], que discute a correspondência entre GLCs e PEGs.

Em seguida, pesquisaram a formalização de extensões de PEGs para dar suporte ao casamento de listas [**?**] e ao uso de regras recursivas à esquerda [**?**]. A pesquisa relacionada ao uso de regras recursivas à esquerda em PEGs influenciou a implementação de vários parsers, tais como IronMeta [**?**] e LPegLJ [**?**].

O projeto atual está relacionado à pesquisa que busca explorar extensões de PEGs que oferecem maior suporte ao relato e à recuperação de erros. Como resultado parcial dessa pesquisa, temos a publicação de artigos científicos [**?**, **?**, **?**], o desenvolvimento da ferramenta LPegLabel [**?**], que possui mais de 5000 downloads, e dois projetos relacionados ao Google Summer of Code [**?**, **?**].

As pesquisas mencionadas anteriormente também contaram com a colaboração de pesquisadores da PUC-Rio e da PUC-PR, que podem eventualmente também coloborarem com este projeto.

13 Infraestrutura

As instituições envolvidas no projeto possuem uma infraestrutura adequada para a execução do mesmo.

A Escola de Ciências e Tecnologia (ECT) da UFRN possui laboratórios e salas para uso exclusivo de professores e estudantes que realizam pesquisa.

Além disso, conta com uma equipe própria de TI que pode dar suporte ao desenvolvimento do projeto.

14 Orçamento Detalhado

A tabela 1 apresenta o orçamento detalhado do projeto para cada ano. O orçamento total do projeto é de R$ 29.980.

Item	Ano 1	Ano 2	Ano 3
Computador	R$ 5.000	R$ 0	R$ 0
Material Bibliográfico	R$ 1.000	R$ 0	R$ 0
Material de Consumo	R$ 200	R$ 200	R$ 200
Passagens Nacionais	R$ 1.500	R$ 1.500	R$ 1.500
Passagens Internacionais	R$ 0	R$ 4.000	R$ 4000
Diárias Nacionais	R$ 960	R$ 960	R$ 960
Diárias Internacionais	R$ 0	R$ 4.000	R$ 4000
Total	R$ 8.660	R$ 10.660	R$ 10.660

Tabela 1: Itens de Custeio

O orçamento prevê 3 viagens nacionais durante o projeto, que incluem viagens para realizar visitas técnicas e para participar de congressos nacionais. Para cada viagem foi estimado um custo de R$ 1.500 para as passagens e de R$ 960 para as diárias.

O orçamento também prevê 2 viagens internacionais para eventos científicos. Para cada viagem foi estimado um custo de R$ 4.000 para a passagem e de R$ 4.000 para as diárias.

Durante o primeiro ano do projeto a UFRN custeará o pagamento de uma bolsa de Iniciação Científica (IC), que será alocada para o aluno Gilney. Esperamos que essa bolsa seja renovada para o segundo ano do projeto.

Referências

[1] OECD/IEA. More data less energy—Making network standby more efficient in billions of connected devices. Technical report, International Energy Agency, 2014.