Towards Example-Guided Network Synthesis

Grupo 1
Francisco Caeiro, **47823**
Bruno Andrade, **47829**
António Estriga, **47839**

Qual é o problema que os autores tentam resolver?

À medida que a quantidade de DSLs (Domain-Specific Language) tem vindo a aumentar, não existe uma adoção dessas novas DSL, devido à dificuldade de aprendizagem e ao facto das mesmas correrem muitas vezes em redes legacy. Para combater este problema, os autores propõem o Facon, uma ferramenta que gera programas em várias DSLs de forma automática e que satisfaz as condições lógicas da rede, através da análise de exemplos de entrada e saída de dados nos switches.

Este problema é relevante?

Os autores abstraem-se da DSL usada no switch, visto que existem cada vez mais, e focam-se principalmente na sua função. Desta maneira, é possível a utilização de diferentes DSLs, desenhadas com o mesmo objetivo, e obter um programa com as mesmas funções.

Qual é a sua solução? Que novas técnicas foram usadas?

A solução proposta foi a criação de uma ferramenta que utiliza lógica matemática para gerar e validar um programa que realiza as mesmas tarefas que uma DSL escolhida. Os autores utilizam relações e cláusulas normais e de Horn para poder gerar as regras necessárias à criação do programa.

A construção do programa é feita em três fases. Na primeira fase, é construído um conjunto de regras iniciais, utilizando relações e variáveis. Estas regras surgem de relações existentes no input, tais como fwd{p1,p2} com as variáveis (p1, p2); esta relação significa que qualquer pacote que chegue a p1 será encaminhado para p2. A ferramenta irá então criar uma regra, fazendo iterações sobre possíveis variáveis e relações, de maneira a representar o intuito do input e output. Na segunda fase, é construído/compilado o programa em si de acordo com as regras feitas anteriormente. Na terceira fase, são realizados testes para garantir que o programa satisfaz todos os inputs/outputs dados. Caso isto não se verifique, volta-se para a fase dois e constrói-se um novo programa candidato.

Quando o programa satisfazer os inputs/outputs, este é refinado em mais três passos. No primeiro passo é feita uma refinação a cada relação, iterando e substituindo as variáveis por nomes (tal como por exemplo: link (switch1, switch2) => link (X,Y)). Isto pode originar regras recursivas, sendo este problema resolvido adicionando o output produzido da nova regra ao input da nova iteração. No segundo passo, as relações já refinadas são agrupadas para construir uma *NDLog rule*. Para criar a NDLog rule, enumeram diferentes partições da relação (link) e agrupam os nomes em vários sub-conjuntos. Desta maneira, evita-se que várias variáveis pertençam a vários sub-conjuntos, removendo regras semanticamente

equivalentes. No terceiro passo, é, por fim, encontrado o conjunto com o menor número de regras que fazem um programa correto, geranado este o programa final.

Como é que se destaca de trabalhos anteriores?

Os trabalhos anteriores não geram programas da mesma maneira que o Facom. Existem ferramentas que compilam políticas de alto nível da rede em configurações de switch, mas requerem uma especificação formal das políticas. O Facon ultrapassa isto ao não necessitar de especificação formal.

Também é de mencionar que existem outras ferramentas capazes de localizar bugs nos programas, algo que o Facon não trata. Se o Facon conseguir criar um programa, não é garantido que este não tenha erros, pois, como é baseado em linguagem declarativa, podem ser introduzidos erros na declaração do input.

Esta ferramenta pode ainda ajudar na criação de novas DSLs, testar a exatidão das mesmas e ajudar na migração de uma DSL para outra.

Quais são os pontos mais fortes deste artigo? E os seus pontos fracos?

De todos os artigos que nos foram atribuídos, quer nesta cadeira que noutras, foi de longe o mais curto com apenas 6 páginas. No entanto, apesar de rápida leitura, deixa de parte explicações mais detalhadas, o que é um aspeto negativo.

Consideramos ainda que o hardware em que foi testado não é apropriado para o mesmo, sendo este o fator mais importante na demonstração de resultados. Este fator pode fazer com que exista uma variação de resultados do mundo real. Os testes deveriam ter sido realizados em ambientes reais, numa rede real.

No entanto, a possibilidade de utilização de qualquer DSL permite uma maior capacidade de abstração da linguagem e focar na lógica da rede.

Como seria uma extensão deste trabalho?

Como é referido no artigo, uma extensão deste trabalho seria o desenvolvimento de geração/recolha de exemplos de input/output de maneira automática, o que facilitaria bastante o processo, em contraste com a recolha manual de exemplos. Esta recolha poderia vir diretamente dos switches e, sendo aplicada ao longo do tempo, ser usada como um modo de aprendizagem contínua. Outra possibilidade de estudo é a obtenção de melhores resultados de desempenho no algoritmo de criação do programa, permitindo uma maior escalabilidade. Outras extensões também são sugeridas, como a expansão para o uso de outros tipos de linguagens.