Introdução

Motivação

"Optimization and constraint programming are beginning to converge, despite their very different origins. Optimization is primarily associated with mathematics and engineering, while constraint programming developed much more recently in the computer science and artificial intelligence communities. The two fields evolved more or less independently until a very few years ago. Yet they have much in common and are applied to many of the same problems. Both have enjoyed considerable commercial success. Most importantly for present purposes, they have complementary strengths, and the last few years have seen growing efforts to combine them. Constraint programming, for example, offers a more flexible modeling framework than mathematical programming. It not only permits more succinct models, but the models allow one to exploit problem structure and direct the search. It relies on such logic-based methods as domain reduction and constraint propagation to accelerate the search. Conversely, optimization brings to the table a number of specialized techniques for highly structured problem classes, such as linear programming problems or matching problems. It also provides a wide range of relaxations, which are often indispensable for proving optimality." – J.N.Hooker

Notação

Ao decorrer do presente texto, usaremos as seguintes convenções:

- Palavras em língua estrangeira estarão *like this*, exceto quando explicitamente dito o contrário;
- A eventual definição de algum termo ou expressão terá o termo ou definição **assim**, com o objetivo de facilitar sua localização; **assim**
- Quando quisermos destacar algum termo por qualquer outro motivo, ele estará *assim*;
- Para trechos de código escritos no meio do texto, será usada `esta fonte`.

Documentação de códigos

As observações nesta subseção servem mais como referência e seriam melhor apreciadas depois de lidos os primeiros capítulos (em particular, o capítulo inicial).

Uma particularidade do Prolog é que, no caso geral, a mesma variável pode ser tanto "de entrada" quanto "de saída" e, como argumento de um funtor,

pode ser tanto instanciada como não instanciada. No entanto, com alguma frequência, um programa é otimizado ou construído para apenas um dos casos. Isto é, frequentemente espera-se que uma variável já esteja instanciada ao ser associada a um funtor (ou o contrário, espera-se que ela não esteja instanciada), ou que uma variável só sirva como variável de saída.

Dada a forma como programas lógicos são escritos, esses casos podem não ser claros e deseja-se que sejam documentados. Para tanto, usaremos a convenção de, na documentação do funtor ou restrição (quando existente, geralmente na forma de um comentário logo antes do funtor ou restrição), indicar os seguintes modos:

- "Variáveis de entrada", isto é, unificadas a outro termo do programa, serão indicadas por um prefixo "+" (como em `f(+X)`);
- Exceto quando são esperadas estarem unificadas com termos base, quando serão indicadas pelo prefixo "++" (como em `f(++X)`);
- "Variáveis de saída", isto é, cuja expectativa é de não estarem unificadas a termo algum, serão indicadas pelo prefixo "-" (como em `f(-X)`);
- Quando não for necessária uma distinção entre "variável de entrada" ou "de saída", a variável será indicada pelo prefixo "?", como o Y em `f(++X, ?Y, -Z)`.

Adicionalmente, o ECL^iPS^edisponibiliza `comment/2`, muito útil tanto para a documentação do código quanto para, possível auxílio do compilador. Não usaremos essa funcionalidade neste texto, por questões de claridade, mas sua utilização é incentivada para códigos "em produção" (consulte o manual para detalhes, assim como para obter um *style guide*, que pode ser de interesse).

Organização

Nosso objetivo principal aqui é introduzir o paradigma de programação por restrições como uma extensão natural do paradigma de programação lógica, com um foco em resolução de problemas, em particular problemas de otimização. Para isso, começamos do básico, tentando dar uma justificativa às escolhas feitas no texto e usando, para isso, de exemplos e código da bibliografia. Caso seja encontrado qualquer irregularidade, erro, ou queira fazer alguma observação, sinta-se livre para enviar um e-mail a ericlesaquileslima at gmail dot com, ou contatá-lo pelo meio que te for mais conveniente.

O texto pode ser razoavelmente bem dividido em três partes:

- Capítulos do 0 ao 6: Desenvolvimento de conceitos básicos e introdução a Programação Lógica, com alguns exemplos;

- Capítulos 7 ao 11: Desenvolvimento de alguns conceitos básicos de programação por restrições, com alguns exemplos;
- Capítulos 8 adiante: Desenvolvimento de mais conceitos de programação por restrições, com um maior foco em resolução de problemas e otimização.

Vale notar que, apesar de usarmos nos exemplos, principalmente, Prolog e ECL^iPS^ee desenvolvermos no texto os conceitos básicos necessários para sua compreensão, este texto não pretende substituir um manual ou qualquer outro texto feito especificamente para esses sistemas.

Agradecimentos

Isto que está lendo provavelmente não existiria não fosse o auxílio proveniente do Programa Unificado de Bolsas da USP e aos incentivos da Marina Andretta, à época minha orientadora, que resolvou apoiar e fazer o que pôde para ajudar o no desenvolvimento projeto. A eles, só tenho gratidão a expressar.
– Éricles Lima