1 Programas como árvores

Será conveniente termos uma interpretação mais visual de programas lógicos. Isso nos dará uma ideia melhor de em qual ordem ocorrem as tentativas de unificações e como lidar com elas. Considere o seguinte programa, representando um pedaço da árvore genealógica da dinastia Julio-Claudiana, junto com a relação ancestral[1]:

```
filhx(julia_caesaris, augustus).
filhx(julia_caesaris, scribonia).

filhx(agrippina, marcus_vipsanius).
filhx(agrippina, julia_caesaris).

filhx(caius_caesar, agrippina).    # Caligula
filhx(caius_caesar, germanicus).

filhx(julia_drusilla, caius_caesar).
filhx(julia_drusilla, caesonia).

filhx(nero, agrippina).           # Nero
filhx(nero, gnaeus_ahenobarbus).

ancestral(A, B) := filhx(B, A), !.
ancestral(A, B) := filhx(B, C), ancestral(A,C).
```

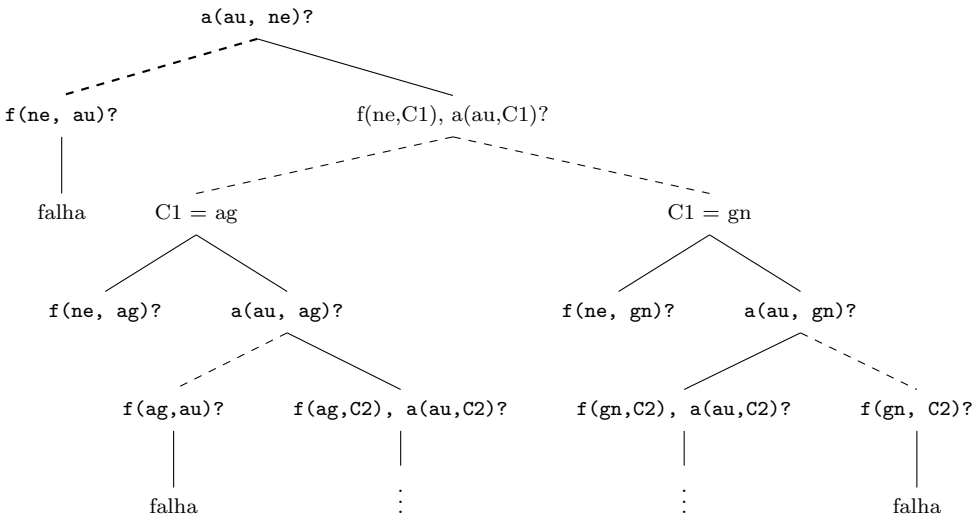
Por conveniência, no lugar das relações filhx/2 e ancestral/2 e dos nomes mostrados acima, usaremos os mostrados abaixo:

```
f(ju, au).
f(ju, sc).

f(ag, ma).
f(ag, ju).

f(ca, ag).
f(ca, ge).

f(ju_d, ca).
f(ju_d, cae).
```

[1] *"It is, of course, obvious at once that 'ancestor' must be capable of definition in terms of 'parent', but until Frege developed his generalised theory of induction, no one could have defined 'ancestor' precisely in terms of 'parent.' "* — Bertrand Russel,*Introduction to Mathematical Philosophy.* Para sermos honestos: na verdade, ele se referia a uma versão mais geral dessa relação do que a apresentada aqui. Voltaremos a esse tema depois.

```
f(ne, ag).
f(ne, gn).

a(A, B) := f(B, A), !.
a(A, B) := f(B, C), a(A,C).
```

O goal `a(au, ne)?` define implicitamente uma árvore, que começa da seguinte forma:



A árvore inteira não será colocada por questões de espaço, mas continua de maneira semelhante. Os ramos tracejados indicam um "ou" lógico, enquanto que os sólidos indicam um "e" lógico. O que isso significa é que falha em um dos ramos dos arcos sólidos indica que todo o arco é falho, mas uma falha em um ramo tracejado só afeta o ramo tracejado.

A execução de um programa lógico consiste em uma busca, em uma árvore como essa, por uma folha de "sucesso". Se, ao invés de "sucesso" é encontrada "falha", é feito o *backtracking*. Quando alguém fala sobre um "modelo computacional de programa lógico", fala essencialmente sobre como travessar essa árvore e como fazer esse *backtracking*. Como pode ver, ela pode crescer muito em largura a cada nível (não cresceu tanto no exemplo acima porque colocamos uma quantidade reduzida de substituições). Por isso, é compreensível que prefiramos uma busca em profundidade do que em largura. Isso pode, é claro, levar a problemas técnicos e filosóficos, já que impõe uma escolha arbitrária sobre a ordem de avaliação das cláusulas e dos termos de cada cláusula, o que pode levar a comportamentos inesperados ao programador desatento a esse modelo[2].

[2]Convém lembrar que não estamos lidando com programação paralela. No modelo de

É claro, dizer apenas que a busca é em profundidade não é o suficiente: precisamos dizer que ela é em profundidade e à esquerda.

Leituras adicionais

[1] Russell, Bertrand (1919), *Introduction to Mathematical Philosophy, George Allen and Unwin, London, UK. Reprinted, John G. Slater (intro.), Routledge, London, UK, 1993*
Esse livro está datado em alguns pontos, mas permanece interessante. Está gratuitamente disponível (em inglês) em [http://people.umass.edu/klement/russell-imp.html].

programação paralela a avaliação pode ocorrer de maneira diferente