8
Conclusão

Este trabalho se prestou à consecução de um objetivo bastante específico: o desenvolvimento de uma definição de verdade baseada nas intuições russellianas acerca desse conceito, que fosse materialmente adequada e que pudesse ser aplicada de modo consistente – até onde isso pode ser garantido – a linguagens semanticamente fechadas. Tão específico como esse objetivo é, não se fazem necessárias muitas considerações para averiguar se ele foi mesmo alcançado. E de fato, no final do capítulo 5 mostramos que o objetivo em questão foi realmente alcançado – a despeito de algumas limitações que a teoria que resultou de nossos esforços neste trabalho possui – visto que nossa definição de verdade satisfaz as condições de adequação que estipulamos para ela.

Por essa razão, para concluir este nosso trabalho, vamos apenas fazer duas coisas que nos parecem apropriadas aqui: vamos apontar para alguns possíveis desdobramentos do trabalho que desenvolvemos, mostrando algumas direções para as quais o tipo de pesquisa realizado no desenvolvimento deste trabalho poderia prosseguir, e vamos mencionar uma possível aplicação que o trabalho que realizamos pode ter, pois embora este trabalho tenha uma natureza predominantemente teórica, e não necessite em princípio de ser passível de nenhum tipo de aplicação para ser considerado válido de algum modo, parece-nos não obstante que ele possui um possível campo de aplicação.

Vamos começar pelos possíveis desdobramentos que pode ter a pesquisa realizada para o desenvolvimento deste trabalho. Parece-nos que pode haver desdobramentos em pelo menos três direções. Em primeiro lugar, a definição de verdade que desenvolvemos pode ser *aperfeiçoada*. De fato, no final do quinto capítulo apontamos para algumas limitações de nossa teoria da verdade, como por exemplo o problema da insuficiência de nossas contrapartes formais dos contextos para a determinação dos aspectos do contexto de uma sentença que são relevantes para a determinação de se uma sentença expressa ou não uma proposição em tal contexto e, em caso positivo, para a determinação do valor de verdade da proposição em questão. Assim, como é evidente que seria desejável obter uma teoria da verdade que satisfizesse as mesmas condições de adequação que nossa

teoria satisfaz, mas que ao mesmo tempo não apresentasse as limitações que ela possui, uma direção na qual um trabalho como o que realizamos poderia prosseguir seria obviamente tentar obter uma teoria da verdade com tais características. Como resultado de um trabalho nessa direção poder-se-ia tanto obter um resultado positivo, caso se conseguisse obter uma definição de verdade do tipo mencionado, ou resultados negativos não menos relevantes, caso a pesquisa nessa direção acabasse levando à descoberta de que dificuldades como essas com contextos apontam para a existência de elementos da linguagem natural que não podem ser capturados por formalismos. Nesse caso, teríamos uma comprovação para a tese de Tarski, exposta na primeira seção do 'The concept of truth in formalized languages', de que não é possível fornecer uma definição de verdade rigorosa cujo campo de aplicação seja vasto o bastante para abarcar as linguagens naturais.

Em segundo lugar, nossa teoria da verdade poderia ser *estendida*. Como mencionamos também no final do capítulo 5, os métodos que utilizamos para a edificação de uma definição de verdade aplicável a linguagens semanticamente fechadas podem muito bem ser utilizados para o desenvolvimento de definições com campos de aplicação igualmente vastos de outros conceitos semânticos, tais como 'define', 'expressa', 'satisfaz', dentre outros. Desse modo, como também mencionamos no capítulo 5, a nossa teoria da verdade poderia ser estendida na direção de uma teoria da semântica em geral, aplicável a linguagens semanticamente fechadas construídas a partir de linguagens de 1ª ordem, ou seja, as linguagens L*, porém enriquecidas com outras constantes predicativas, além de V, representando outros predicados semânticos além do predicado-verdade.

Por fim, como mencionamos no capítulo 1, uma terceira direção para a qual o trabalho que realizamos aqui poderia prosseguir seria a tentativa de desenvolver uma teoria da verdade capaz de satisfazer as condições de adequação que impusemos sobre a nossa teoria, e que tomasse o predicado-verdade como um predicado de proferimentos. Como a teoria da verdade de Kripke é uma teoria com as características mencionadas que aplica o predicado-verdade às sentenças, e a teoria que desenvolvemos neste trabalho é uma teoria desse tipo que aplica o predicado-verdade às proposições, uma teoria que satisfizesse os mesmos *desiderata* e que aplicasse o predicado-verdade aos proferimentos completaria o quadro, e uma análise comparativa dessas teorias certamente mostraria quais as

vantagens formais – uma vez que, como comentamos no capítulo 1, nos parece que não pode haver vantagens de outro tipo nesse caso – de se tomar sentenças, proposições ou proferimentos como portadores de verdade.

Isso é o que temos a dizer sobre os desdobramentos que o trabalho que realizamos aqui pode vir a ter. Agora, um breve comentário sobre uma possível aplicação para este trabalho. O teste do valor de verdade de uma condição é um fator importante na programação de computadores, como bem sabe qualquer um que se tenha dedicado minimamente à programação. Quando criamos um programa de computador, de fato, o que fazemos essencialmente é passar instruções à máquina sobre o que ela deve fazer caso se verifiquem determinadas condições. Assim, se quisermos que a máquina nos dê a área de uma circunferência com raio *r* se lhe dermos um número real positivo como valor de *r*, e que nos dê uma mensagem de erro caso lhe dermos qualquer outra coisa como valor de *r*, vamos dizer isso a ela por meio de linhas de algoritmo do tipo seguinte:

se r > 0 então c = π * r^2 e imprima "A área é" c;

senão imprima "Você digitou uma expressão inválida para esta operação".

Linguagens de programação de alto nível em uso tais como PASCAL e C utilizam linhas de programa do tipo acima caso se quiser obter da máquina a execução do procedimento que descrevemos. Mas nesse caso, para saber como proceder, a máquina deverá ser capaz de determinar o valor de verdade da condição 'r > 0'. Os processadores que temos hoje em dia são mesmo muito eficazes em fazer isso, depois que a instrução recebida foi convertida por um compilador para linguagem de máquina, isto é, para código binário, mas as pessoas que desenvolveram os processadores (ou os compiladores, de qualquer modo) precisaram fabricá-los de um modo tal que eles dessem as respostas corretas em cada caso. Assim, o projetista de um processador precisou desenvolvê-lo de modo que, se a condição a avaliar fosse a disjunção 'r > 0 ou r < 100', por exemplo, ele fosse capaz de avaliá-la como verdadeira caso ao menos um dos dois disjuntos tivesse sido antes avaliado como verdadeiro.

A maneira como os processadores fazem isso utilizando código binário é, obviamente, algo que interessa à engenharia elétrica e à ciência da computação, mas para saber como projetar seus processadores para dar determinadas respostas em determinadas situações, o projetista precisa apelar a conhecimentos exteriores

a essas áreas, como no caso mencionado acima da condição disjuntiva. Esse é um caso simples, e não é necessário saber muita lógica para saber como projetar um processador para dar as respostas corretas nesse caso. Mas objetivos mais arrojados em ciência da computação levaram ao desenvolvimento de linguagens de programação capazes de incluir condições bem mais complexas que a máquina deve saber avaliar. Para finalidades relativas à inteligência artificial, por exemplo, foi desenvolvida a linguagem PROLOG, cuja base é o cálculo de predicados de 1ª ordem. Desse modo, ao lidar com programas escritos nessa linguagem, os compiladores PROLOG devem saber explicar ao processador em código binário como avaliar condições como '$\exists x$ (Fx & ~Gx) v $\forall x$~Hx'. Ora, é claro que para que o processador possa avaliar uma condição como essa, isto é, para que ele possa executar as instruções certas nos casos certos, o seu projetista – ou o desenvolvedor do compilador – deverá saber quais as condições em que essa fórmula é verdadeira, e para tanto ele deverá ter conhecimento da semântica tarskiana para as linguagens de 1ª ordem.

Agora, um dos principais campos específicos de aplicação de linguagens de programação como PROLOG, em inteligência artificial, é a área de processamento da linguagem natural, cujo objetivo máximo é a obtenção de máquinas capazes de processar a linguagem natural, isto é, capazes de entender instruções dadas em linguagem natural e de fornecer respostas em linguagem natural. E algo que todo usuário de computadores pode facilmente notar é que nossas máquinas e programas atuais não são particularmente hábeis no que se refere a processar a linguagem natural no sentido mencionado.

De fato, linguagens de programação como PROLOG podem fazer avançar pouco a criação de programas capazes de processar a linguagem natural por uma razão muito conhecida dos lógicos: as linguagens formais de 1ª ordem – e como dissemos acima a base de PROLOG é o cálculo de predicados de 1ª ordem – capturam um fragmento bastante restrito da linguagem natural. Para capturar fragmentos mais amplos das línguas naturais (ou toda uma língua natural, se isso for possível) é necessário lançar mãos de formalismos mais poderosos, e para que se possa desenvolver linguagens de programação baseadas nesses formalismos, é preciso que os mesmos estejam acompanhados de definições de verdade com um campo de aplicação vasto o suficiente para abarcar tais formalismos, de modo que

haja como avaliar de modo preciso o valor de verdade de uma condição escrita em um desses formalismos, e então executar as instruções relacionadas.

Enfim, é óbvio que para os objetivos da área de processamento da linguagem natural o mais interessante seria a obtenção de uma teoria da verdade capaz de avaliar de modo preciso uma condição escrita *em* linguagem natural. Como não sabemos se isso é possível – e de todo modo enquanto esse objetivo não é alcançado – definições de verdade com um campo de aplicação vasto, capazes de lidar, como faz nossa definição, com linguagens semanticamente fechadas (o que é o caso das linguagens naturais), com sentenças sem sentido, com sentenças paradoxais e, ainda que precariamente, com a variação de valor de verdade em contextos, certamente constituem material teórico do interesse de criadores de linguagens de programação e, mais remotamente, de projetistas de *hardware*, que tenham entre seus objetivos o desenvolvimento de programas e máquinas capazes de processar a linguagem natural.