3ª PARTE

UMA ANÁLISE INTERNALISTA DOS NOMES PRÓPRIOS

Capítulo VI

A função operacional: a função básica dos nomes próprios

6.1. Considerações preliminares

Minha visão da linguagem é a de uma ferramenta para o intercâmbio de informações. No meu modo de ver, as expressões da linguagem são códigos que fornecem informação para sistemas de decodificação apropriados. Esses sistemas integram as mentes dos usuários da linguagem. Dessa forma, fica claro que minha visão da linguagem é francamente internalista. Acredito que nosso uso da linguagem só pode ser explicado satisfatoriamente se incluirmos na explicação uma análise de como a mente humana torna esse uso possível.

Integrada à minha visão da linguagem há, portanto, uma concepção de *mente*. Essa concepção não é outra senão a concepção computacional de mente que é apresentada no capítulo 4. Para mim, como para todos os cognitivistas, a mente consiste em um conjunto de sistemas de computação e representação que processam informação ou contribuem de alguma forma para esse processamento. Essa parece ser a única concepção de mente que nos permite explicar como é possível aos seres humanos desenvolverem e usarem a linguagem.

Neste capítulo, vou apresentar minha tese propriamente dita, a tese de que a função básica dos nomes próprios é a mesma função básica de todas as expressões da linguagem, a saber, a função operacional, a função de código. Isso equivale a dizer que nomes próprios também são informativos, que sua principal função é fornecer informação para nossos sistemas de processamento de linguagem.

Pelo que é dito no parágrafo acima, fica claro que minha tese sobre os nomes próprios é tributária da minha visão internalista sobre a linguagem. É de se esperar, portanto, que um detalhamento dessa visão seja requerido para esclarecer a tese. É o que efetivamente faço neste capítulo. Primeiramente apresento minha concepção a respeito da função operacional da linguagem em geral e a partir daí começo a delimitar o problema e mostrar que os nomes próprios não podem ser considerados uma exceção no que diz respeito à capacidade de exercer essa função. Além disso, proponho também um modelo de processamento dos nomes próprios que servirá para explicitar ainda mais meu ponto de vista. É preciso ver,

porém, que minha tese a respeito da função básica dos nomes próprios não depende do meu modelo de processamento dos nomes próprios para ser estabelecida. A tese é estabelecida a partir dos argumentos que ofereço para evidenciar que todas as expressões da linguagem têm uma função operacional, inclusive os nomes próprios. Ainda assim, o modelo tem uma função importante. Se nomes próprios realmente veiculam um tipo de informação, é interessante sugerir uma hipótese sobre o modo como essa informação é processada e sobre os resultados desse processamento. O modelo é essa hipótese. É nesse sentido que ele é importante. Embora ele não possa ser considerado uma demonstração de como nomes próprios são processados pela mente humana, ele pode e deve ser visto como uma hipótese plausível de como isso acontece. Isso certamente não é essencial para o estabelecimento da minha tese, mas é importante para iluminar o quadro geral.

O capítulo está organizado da seguinte forma: Na seção 6.2., trato de esclarecer minha noção de *informação*, que é indiscutivelmente a noção mais importante pressuposta pela minha argumentação a respeito da função operacional da linguagem. Uma vez que essa noção é definida, torna-se possível explicar em que sentido nomes próprios são informativos. Na seção seguinte, apresento os devidos argumentos a favor da minha tese, ou seja, procuro estabelecer que a função operacional é a função mais básica da linguagem, em geral, e dos nomes próprios, em particular. Por fim, na seção 6.4, apresento meu modelo de processamento de nomes próprios. Esse modelo está baseado em um programa que chamo de “ProgX”, o qual é concebido para responder a pergunta “quem é X?”, sendo X um nome próprio qualquer. O programa não é realmente escrito, ou seja, o que apresento não é um programa escrito em linguagem de programação e pronto para ser compilado e executado. O que apresento é uma descrição geral de ProgX e uma argumentação de que ele constitui um modelo de como nomes próprios são processados pela mente.

6.2. A noção de *informação*

A noção de *informação* é geralmente definida com base nas noções de *dado* e *significado*. Luciano Floridi, em seu artigo *Semantic Conceptions of Information*, escrito para a Stanford Encyclopedia of Philosophy, esclarece que, na maioria dos

campos em que a noção de *informação* é utilizada, adota-se uma definição geral de informação que diz basicamente que

Def.: Informação = dados + significado

O que é o mesmo que dizer que uma informação é constituída de dados significativos, isto é, dados interpretáveis.

Essa definição tem muitos aspectos positivos. O maior deles é que ela realmente chega muito próximo de captar o que a palavra "informação" expressa em seu uso científico e filosófico. Todavia, encontro nela pelo menos dois problemas.

O primeiro tem a ver com o uso da noção de *significado* na definição. Tal uso é problemático porque a noção de *informação* me parece ser mais geral e mais básica do que a noção de *significado*. Ela é mais geral no sentido de que, no meu modo de ver, há casos em que poderíamos dizer que uma informação é fornecida, mas não um significado, pelo menos não se a palavra "significado" é usada em sua acepção comum. E, como tentarei demonstrar no próximo capítulo, é mais básica no sentido de que a maioria dos fenômenos semânticos podem ser descritos como fenômenos informacionais. Por isso mesmo, seria mais adequado usar a noção de *informação* para esclarecer a de *significado* do que fazer o contrário.

O outro problema que encontro é que a definição parece dizer que a informação *é* o dado, em vez de dizer que a informação *está contida* no dado. Para o uso filosófico que quero fazer da palavra "informação", é melhor assumir a imaterialidade da informação. É melhor assumir que uma informação é uma coisa que sempre está contida em um meio material, mas de fato, não pode ser identificada com esse meio. A mesma informação pode estar contida em diferentes meios. Por exemplo, se considerarmos como dados linguísticos os seguintes enunciados: 1. "está chovendo" e 2. "it's raining", parece razoável assumir que esses diferentes dados veiculam a mesma informação para alguém que entenda português e inglês. Dessa forma, assumimos que a informação é algo diferente do meio material na qual ela é codificada. Com isso, concordamos com Norbert Wiener quando ele diz: "Informação é informação, não matéria nem

energia. Nenhum materialismo que não admite isso pode sobreviver nos dias atuais" (citado em GARDNER, 2003: 36).

Feitas essas ressalvas, gostaria de propor outra definição de "informação". A definição ainda dependerá da noção de dado, mas, no lugar da noção de *significado*, utilizará a noção de *sistema de decodificação*. A definição é a seguinte:

Def.: Um dado A fornece informação para um sistema de decodificação S se e somente se S tem uma mudança de estado quando recebe A.

De acordo com essa definição, uma informação é algo que é fornecido por um dado a um sistema de decodificação. Um dado informativo é o que chamaremos de "código". É preciso chamar atenção para alguns pontos da definição.

Em primeiro lugar, é necessário definir a noção de *dado*. A definição oferecida por Floridi declara que "um dado é um fato putativo relativo a alguma diferença ou carência de uniformidade em um contexto" (FLORIDI, op. cit.). Embora essa definição seja correta, ela é um tanto obscura. Uma definição mais simples e igualmente correta seria:

Def.: Um dado é um aspecto que se diferencia em um contexto [1].

Pode-se postular que essa diferenciação existe na realidade independentemente de ser detectada por um ser vivo ou por uma máquina, ou pode-se considerar que ela só existe enquanto diferença detectável. Nesta última perspectiva, qualquer coisa ou aspecto que um indivíduo ou uma máquina possa distinguir na realidade é um dado. Na primeira perspectiva, até as coisas que não podem ser distinguidas, mas que se distinguem das outras por um tipo de diferença substantiva (de substância), são dados. Em todo caso, a definição de *dado* independe da perspectiva adotada. E, consequentemente, o mesmo vale para a definição de *informação*.

[1] A noção de *aspecto* não é definida aqui. Em todo caso, gostaria que ela fosse entendida em seu sentido coloquial.

Em segundo lugar, é preciso esclarecer a noção de *sistema de decodificação* [2]. Um sistema de decodificação é qualquer sistema que implementa uma função de transição de estados. Um estado é cada uma das configurações (conjuntos de parâmetros) que um sistema apresenta ao longo do tempo, e uma função de transição de estados é uma regra que diz o que deve acontecer para o sistema passar de um estado para outro. Alguns exemplos de sistemas de decodificação são: programas, autômatos (como um editor de texto), sistemas químicos (como o procedimento usado para testar um tipo sanguíneo), sistemas biológicos (como o sistema celular que decodifica a informação contida em uma cadeia de DNA), dispositivos mecânicos (como uma fechadura) etc. Por esses exemplos, vê-se que não é correto supor que o processamento de informação se dá apenas em um nível abstrato e depende sempre de algoritmos elaborados, como em programas de computadores. Com efeito, é perfeitamente apropriado dizer que uma fechadura processa a informação fornecida pela chave girando, que a flauta processa a informação fornecida pelo sopro e pela digitação etc. O processamento de informação não pressupõe linguagem. Na verdade, o que pretendo mostrar aqui é exatamente o contrário: o uso da linguagem pressupõe processamento de informação.

Em terceiro lugar, deve-se notar que a noção de *informação* não é definida simplesmente como uma propriedade de um dado. Ela é definida com base numa relação entre um dado e um sistema de decodificação. A informação fornecida por um dado é sempre uma informação para um sistema de decodificação específico. Aqui identificamos mais uma das características essenciais da informação: a informação é específica, vale dizer, direcionada a um sistema. Isso significa que o mesmo dado pode fornecer uma informação valiosa para um sistema de decodificação S e nenhuma informação para um sistema de decodificação S' e, inversamente, um sistema de decodificação pode ser perfeitamente adequado para interpretar certos dados e ser totalmente inútil para decodificar outros. Assim, se a especificidade da informação não for respeitada, nem o dado fornecerá informação para o sistema (é por isso que a chave do meu

[2] De fato, em vez de usar a expressão "sistema de decodificação" poderia usar simplesmente a palavra "sistema", mas para isso teríamos que interpretar a palavra de acordo com um sentido muito estrito. Como a palavra tem outros sentidos, inclusive um sentido muito próprio do contexto filosófico (e.g., "Gilles Deleuze não tem um sistema"), acredito ser melhor usar a expressão "sistema de decodificação".

carro não serve para ligar outros carros), nem o sistema poderá ler a informação apropriada a partir do dado (é por isso que o meu carro só pode ser ligado com a minha chave – pelo menos eu espero). De fato, há muitos exemplos dessa especificidade da informação: um eletroencefalograma contém uma informação que só pode ser decodificada por um médico especialista, um *cd* contém uma informação que só pode ser lida por um *cd player*, a versão original de *Ana Karenina* contém uma informação que só pode ser processada por alguém que sabe ler russo etc. Note, porém, que não é necessário que um dado contenha só um tipo de informação. A informação é específica, mas o dado não. O mesmo dado A que fornece informação para um sistema S, pode fornecer uma informação diferente para um sistema diferente. Por exemplo, a versão original de *Ana Karenina* fornece informação para um leitor russo, mas também pode fornecer informação para um *scanner* ou para uma máquina fotocopiadora. Em todo caso, as informações são diferentes.

Note também que podemos dizer que um dado A contém informação mesmo que a informação não esteja sendo processada e mesmo que não saibamos como ela poderia ser processada ou por qual sistema. Parece natural admitir, por exemplo, que os hieróglifos nos monumentos egípcios continham informação desde que foram escritos, muito antes portanto da descoberta da pedra de Roseta. É interessante observar que admitimos isso porque os hieróglifos fornecem uma informação prévia mesmo para quem não sabe lê-los. Quem vê a escrita hieroglífica, mesmo que não seja capaz de decifrá-la, é levado a crer que aquilo se trata de um código. É levado a crer que houve um sistema capaz de decodificar esse código e que esse sistema poderia ser redescoberto ou até reinventado. A informação que leva a essa crença parece advir não só das representações pictóricas da escrita hieroglífica, mas principalmente do fato da organização dessas representações sugerir uma estrutura sintática. Assim, de uma forma geral, se um dado aparenta ser informativo, ele já está sendo informativo (embora, frequentemente, a informação que podemos decodificar não é a informação que gostaríamos de poder decodificar).

Depois dessas observações sobre a noção de *informação*, cabem ainda alguns esclarecimentos adicionais sobre o processamento da informação.

Uma coisa que se deve notar é que um dado pode ser inserido em um sistema de decodificação em conjunto com outros. Em tais circunstâncias,

diremos que o sistema recebeu um *pacote de dados*. A informação fornecida por um dado pode variar de acordo com o pacote do qual ele faz parte. Aqui temos um tipo de princípio do contexto da informação. Por exemplo, o som de um apito usado por um árbitro de futebol em uma partida de futebol fornecerá uma informação diferente daquela fornecida pelo apito de um guarda de trânsito em serviço. Às vezes, mesmo um dado que habitualmente não informa nada, ao ser processado em conjunto com outros dados, pode se tornar informativo. Por exemplo, o silêncio pode ser informativo se ele for considerado como a ausência de um barulho habitual. Por exemplo, se eu ligo meu som, aumento o volume e não ouço nada, isso é bastante informativo. Ou meu som está com defeito, ou eu estou surdo.

Outra coisa que deve ser explicada é o que um sistema de decodificação efetivamente faz. Bem, o que ele faz é ler a informação de um dado e executar uma ação. A informação é o gatilho que dispara a ação. Muitas vezes, vários sistemas podem ser combinados em série, de modo que os dados de saída de um servem como dados de entrada do outro. Nesses casos, a ação que os sistemas intermediários executam é uma ação de transformação. A informação recebida pelo sistema é transformada em uma nova informação, e essa informação, por sua vez, será lida por outro sistema. É isso que acontece, por exemplo, quando um pianista lê uma partitura. A informação fornecida pela partitura é lida por um sistema na mente do músico e transformada em informação mecânica, presente nos movimentos que o pianista faz para tocar. Em seguida, essa informação é lida pelo instrumento e transformada em informação sonora; e a informação sonora, por sua vez, é processada pelos sistemas auditivos dos ouvintes, e transformada em informação baseada em sinais elétricos, e assim passará a outros sistemas que a transformarão também.

Muitas vezes, um sistema produz uma nova informação, mas ela não é logo utilizada por outro sistema. Nesses casos, o que o sistema faz é armazenar a informação, inscrevendo-a em um novo meio. É isso que acontece, por exemplo, quando gravamos um *cd* e não o colocamos logo para tocar.

Deve-se notar, como se depreende do exemplo do pianista, que é possível falar de diferentes tipos de informação. Essa classificação é conveniente, no entanto, a rigor, não é a natureza da informação que muda, e sim a natureza do meio que a contém. É o meio que é mecânico, ou sonoro, ou de qualquer outro

tipo, não a informação, pois esta, como vimos anteriormente, não possui uma natureza material. Assim sendo, o que chamo de “informação mecânica” no exemplo do pianista, não é nada mais do que a informação fornecida por um movimento. O que chamo de “informação sonora”, no mesmo exemplo, é tão somente a informação veiculada por um som. E, generalizando, vou dizer que uma informação é do tipo x, se x é um adjetivo referente ao meio que serve de veículo para a informação. Nesse sentido, é possível falar de informação sensorial, informação eletrônica, informação biológica, informação linguística e outras semelhantes.

É importante observar também que, em muitos casos, dois dados lidos pelo mesmo sistema de decodificação podem levar o sistema ao mesmo estado. Por exemplo, em muitos casos, é possível votar em uma reunião levantando um braço ou fazendo um sinal com o polegar, e o voto sempre tem o mesmo valor. Nesses casos, deve-se considerar que, embora os dados sejam materialmente diferentes, a informação fornecida por eles para aquele sistema é exatamente a mesma.

Pode-se pensar que expressões sinônimas são dados linguísticos que fornecem a mesma informação para um usuário da linguagem, mas não é necessariamente assim. Deve-se notar que a relação de sinonímia é uma relação semântica e não pragmática, o que significa que ela pode ser definida ou explicada sem que seja preciso mencionar qualquer suposto efeito das expressões nos falantes. Efetivamente, pode-se dizer que duas expressões R e S são sinônimas se o dicionário diz que elas são sinônimas ou se S é igual a R a não ser por apresentar um sinônimo da expressão T onde R apresenta T. A informação fornecida por uma expressão T, por outro lado, depende do sistema de decodificação que recebe T e depende do contexto em que T é proferido. É fácil exemplificar os dois casos.

Digamos, por exemplo, que o falante A apresente seu filho para o falante B dizendo: “este é meu primogênito”. Se B não conhece a palavra “primogênito”, ele não vai poder decodificar adequadamente o proferimento de A. Se, em contrapartida, A disser “este é meu filho mais velho”, a informação veiculada será decodificada adequadamente. Os dois proferimentos de A são sinônimos, mas não veiculam a mesma informação para B. Isso mostra que a

informação veiculada por expressões sinônimas pode variar de acordo com quem vai decodificar as expressões.

Outrossim, o contexto também pode fazer a informação variar. De fato, a informação veiculada por um proferimento pode mudar radicalmente conforme ele seja interpretado como uma afirmação literal ou como uma ironia. Por exemplo, digamos que um falante C diz de um indivíduo “aquele é nosso amado chefe” e que um falante D diz do mesmo indivíduo “aquele é nosso estimado chefe”, e, além disso, suponhamos que o proferimento de C é interpretado literalmente e que o proferimento de D é interpretado como uma ironia. Nesse caso, podemos admitir que temos duas sentenças sinônimas comunicando informações diametralmente opostas para os seus respectivos ouvintes.

Outro caso de influência do contexto no conteúdo informativo de expressões sinônimas ocorre quando essas expressões são usadas como *definiendum* e *definiens* de uma definição. Quando isso acontece, não é razoável afirmar que as expressões veiculam a mesma informação. Por exemplo, na definição “primogênito é o filho mais velho”, não é razoável pensar que a expressão “primogênito” veicula a mesma informação que a expressão “o filho mais velho”. Parece mais sensato assumir que a definição como um todo funciona como uma instrução para um sistema de decodificação na mente do leitor ou ouvinte. Tal instrução indica que o sistema deve relacionar a palavra “primogênito” com certa informação contida na expressão “o filho mais velho”. Se, na definição, a palavra “primogênito” informasse o mesmo que a expressão “o filho mais velho”, não precisaríamos de uma definição. A definição não diria nada de novo.

Em todos esses casos fica claro que, diferentemente da relação de sinonímia, a veiculação de informação é sempre sensível a fatores pragmáticos, o que implica que ela depende de que um conjunto maior de condições seja satisfeito. A informação fornecida por uma expressão é sempre uma informação fornecida para determinado ouvinte, em determinadas circunstâncias. Mesmos as informações veiculadas por expressões sinônimas podem variar junto com as circunstâncias de seu proferimento e decodificação. Somente quando essas circunstâncias não causam variação na informação veiculada pelas expressões é que é possível dizer que elas veiculam uma mesma informação. Mesmo assim,

ninguém deve pensar que o conteúdo informativo das expressões como um todo seja o mesmo; a informação compartilhada é apenas uma parte desse todo, e toda a informação restante pode divergir. Por exemplo, a informação fonética sempre vai divergir. Também pode haver informações ligadas ao registro linguístico em que uma expressão tipicamente ocorre, e expressões sinônimas também podem divergir em relação a esse tipo de informação, mesmo quando mantêm um núcleo informativo em comum. Por exemplo, a expressão “primogênito” geralmente não ocorre na fala coloquial. Esse fato pode informar algo sobre a educação formal do falante que proferiu a expressão. Por outro lado, se a expressão proferida fosse “filho mais velho”, o ouvinte não teria a mesma informação sobre o falante (pelo menos não com base na expressão proferida). Além disso, é sabido que as palavras podem ser associadas a certas lembranças e a certas emoções. Destarte, mesmo quando duas palavras sinônimas contêm uma informação comum, uma delas pode ser mais informativa que a outra para um determinado ouvinte. Uma palavra pode despertar no ouvinte sensações que a outra palavra não desperta. E esses são apenas alguns exemplos que mostram que há muito mais informação em uma expressão do que aquela que ela compartilha com uma expressão sinônima. Muitos outros exemplos poderiam ser dados.

Para terminar esta seção, gostaria apenas de chamar a atenção para um ponto importante da discussão apresentada até aqui. Nos parágrafos anteriores, falo de certos sistemas da mente como sendo sistemas de decodificação. De fato, como já mencionei no começo desse capítulo, vejo a mente humana como um conjunto de sistemas de computação e representação. Acredito que quando usamos a linguagem, alguns desses sistemas são acionados. Esses sistemas têm a função de decodificar e processar as informações veiculadas pelas expressões da linguagem, inclusive pelos nomes próprios.

Na seção seguinte, recorrerei constantemente a essa ideia para mostrar que a função básica da linguagem é a função operacional, vale dizer, a função de fornecer informação para nossos sistemas internos de decodificação. Mostrarei também que mesmo os nomes próprios têm uma função operacional e que, apesar de muitos filósofos considerarem (ou pressuporem) que a função referencial é sua função mais básica, os nomes só podem desempenhar essa função, ou qualquer outra que seja, se antes desempenharem uma função operacional.

6.3. A função operacional da linguagem e dos nomes próprios

Recentemente li a seguinte frase: “A catedral de São Basílio em Moscou é composta por nove capelas independentes”. Antes de ler essa frase, não sabia que a catedral de São Basílio em Moscou é composta por nove capelas independentes, mas fiquei sabendo logo que acabei de ler.

Em certa ocasião, estava dando carona para uns amigos e a gente se perdeu. Ao ver algumas pessoas na rua, parei para pedir uma orientação. Não me lembro bem do que as pessoas me disseram, mas vou fazer de conta que foi o seguinte: “vá em frente e vire à esquerda na terceira rua, siga até o sinal e vire à direita”. Eu fiz o que eles me disseram e consegui chegar aonde queria.

Também me lembro que uma vez ouvi uma piada sobre dois amigos, um chamado “Joaquim” e outro chamado “Manuel”. Antes de ouvir a piada, eu estava sério e não sentia nenhuma vontade de rir. Depois de ouvi-la, ri tanto que quase perdi o fôlego.

Ao lembrar esses episódios, eu fico a pensar: como é possível que minha ignorância a respeito da composição da catedral de São Basílio tenha se desvanecido por causa de certas marcas em um papel? Como é possível que eu estivesse perdido e, tão logo ouvisse certos sons, achasse meu caminho? Como é possível que minha circunspecção tenha dado lugar a uma sonora gargalhada assim que eu terminei de ouvir certos proferimentos?

Ao pensar nisso, eu sou levado a concluir que aconteceu algo entre o momento em que eu vi a frase sobre a catedral de São Basílio e o momento em que meu conhecimento sobre ela aumentou. Aconteceu algo entre o momento em que ouvi as instruções para me localizar e o momento em que comecei a seguir aquelas instruções [3]. E, por fim, aconteceu algo entre a piada e o riso. O que aconteceu em todos esses casos foi a decodificação de certas informações. De onde vieram essas informações? Obviamente vieram dos proferimentos que eu li ou ouvi. Eu recebi um proferimento, decodifiquei-o e, como resultado dessa decodificação, eu fiquei mais culto, ou achei o caminho para casa, ou ri às bragas soltas. Não posso descrever exatamente o que aconteceu na minha mente durante

[3] Quando estava escrevendo isso, lembrei de uma passagem das *Investigações Filosóficas* que diz o seguinte: ““Entre a ordem e a execução há um abismo. Este tem que ser fechado pela compreensão.” “Somente na compreensão se diz que temos que fazer ISTO. A ordem – é apenas sons, traços de tinta. –”” (WITTGENSTEIN, 1996, § 431).

essa decodificação (embora tenha algumas ideias a respeito), mas não tenho dúvida nenhuma de que ela aconteceu. E sei também que, se ela aconteceu, foi porque os proferimentos que eu li ou ouvi funcionaram como códigos, isto é, como dados carregados com informação. Não vejo como explicar esses fatos de outra forma.

Uma das minhas ideias fundamentais neste trabalho é a de que as expressões da linguagem sempre são usadas como códigos. Ora, para mim, dizer que as expressões da linguagem sempre são usadas como códigos é exatamente o mesmo que dizer que a função básica da linguagem é a função operacional. Quando uso a expressão "função operacional" estou me referindo a essa função de código. A razão de usar essa expressão é para aludir ao fato de que sempre que a linguagem é usada, há sistemas nas mentes dos falantes que são acionados e que realizam certas operações para processar as informações que são fornecidas pelas expressões da linguagem. Poderia usar também a expressão "função informacional", mas assim me arriscaria a causar uma certa confusão, uma vez que alguém poderia pensar que a função informacional é a função de informar, e informar no sentido de relatar fatos através de sentenças declarativas. Obviamente não é essa a ideia. Assim, acho preferível falar de "função operacional".

Parece evidente que a função operacional é a função básica da linguagem. E dizer que essa função é básica significa dizer que ela sempre é exercida quando a linguagem é usada corretamente e que qualquer outra função a pressupõe. De fato, podemos fazer um proferimento para alcançar diferentes objetivos, mas, até onde posso ver, qualquer que seja o objetivo, ele só será alcançado se nosso proferimento puder ser decodificado por um sistema de decodificação adequado. A propósito, deve-se observar que os proferimentos envolvidos em cada um dos exemplos citados acima tinham naturezas muito diversas. Cada um tinha uma finalidade diferente, o que significa que cada um tinha uma função pragmática diferente.

O primeiro era uma sentença descritiva [4], isto é, era uma sentença com a pretensão de descrever a realidade. Acredito que se tratava de uma sentença verdadeira, mas bem que poderia ser uma falsidade. De qualquer forma, sei que

[4] Austin preferia usar a expressão "sentença constativa" em vez de "sentença descritiva". Ele justificava essa preferência dizendo que nem toda sentença verdadeira ou falsa é uma descrição (cf. AUSTIN, 1975: 3). Mesmo considerando apropriada a terminologia de Austin, vou usar a expressão "sentença descritiva" pelo fato de ela ser mais coloquial.

ela provocou certas mudanças no meu conjunto de crenças, e isso aconteceu por causa da informação que ela forneceu para um sistema de decodificação na minha mente. Alguém poderia imaginar que uma informação só foi fornecida nesse caso porque o proferimento que me foi oferecido era uma sentença descritiva. Parece natural que uma sentença descritiva transmita informação sobre o mundo. Isso é verdade, mas não é só porque a sentença veicula informação sobre o mundo que ela é informativa. Há outras informações que são fornecidas pela sentença. Há informações ligadas a cada uma das palavras que a compõem, há informações ligadas a sua estrutura sintática e até informações ligadas ao contexto em que ela é proferida. Todas essas informações são processadas independentemente do fato do proferimento ter algo a dizer sobre o mundo. De fato, os outros exemplos provam que não são apenas as sentenças descritivas que veiculam informação para nossos sistemas internos de decodificação.

O segundo proferimento, por exemplo, era uma instrução complexa. Ele não fazia nenhuma constatação sobre a realidade (embora se apoiasse na pressuposição de que o mundo era caracterizado objetivamente por certos estados de coisas), mas, mesmo assim, é inegável que ele me forneceu um tipo de informação extremamente útil. O que garante que essa informação foi recebida e processada na minha mente é que minha ação foi um tipo de resposta ao proferimento. Depois que ouvi as instruções, fiz exatamente o que fui instruído a fazer, e a consequência disso foi que encontrei o caminho para casa. Parece razoável imaginar que os movimentos que fiz para dirigir o carro foram ordenados por meu cérebro como resultado da decodificação das instruções que recebi.

Da mesma forma, o terceiro proferimento não era composto de sentenças que visavam descrever algum fato do mundo. O objetivo das sentenças era apenas fazer rir. Mesmo assim, para cumprir o objetivo de fazer rir, o proferimento precisava sensibilizar de alguma forma meus sistemas internos de decodificação. Consequentemente, é preciso admitir que o proferimento me abasteceu com algum tipo de informação e que o processamento dessa informação culminou com uma boa gargalhada. Nesse caso, é interessante observar que os nomes “Manuel” e “Joaquim” não foram proferidos com a intenção de referir ninguém. Independentemente disso, na medida em que eles ajudaram a compor o proferimento e contribuíram para o seu efeito cômico, eles foram usados como códigos e, nesse sentido, podem ser considerados informativos. Admito que essa

afirmação de que os nomes são informativos precisa de uma justificação mais criteriosa. E, de fato, ela será dada logo mais.

Em todos esses exemplos, uma coisa fica evidente. Seja qual for a finalidade para a qual o proferimento é designado, para que essa finalidade seja cumprida, é necessário que o proferimento seja processado pela mente em primeiro lugar. Dessa maneira, pode-se concluir que todo proferimento, enquanto cumpre um propósito na comunicação, deve antes de tudo exercer uma função operacional, ou seja, deve fornecer informações que acionam certas operações em meus sistemas internos de decodificação. Se se trata de uma sentença declarativa, ela só pode me dar algum conhecimento se me fornecer informação. Se se trata de uma ordem ou instrução, ela só pode ser executada, seja por palavras, seja por ações, se me fornecer informação. Se se trata de uma piada, ela só pode me fazer rir se me fornecer informação. Efetivamente, se os proferimentos dos exemplos que eu dei me fossem oferecidos em mandarim, eu ainda seria ignorante sobre a composição da catedral de São Basílio, teria tido uma dificuldade consideravelmente maior para me localizar e, muito provavelmente, teria continuado circunspecto. As coisas se passariam dessa forma porque eu não tenho um sistema de decodificação de mandarim. Assim, fica claro que os objetivos de um proferimento só são realizados se, em primeiro lugar, ele puder ser decodificado. Em outras palavras, em todo uso eficiente da linguagem, a função operacional é pressuposta.

Imagino que alguém poderia argumentar que meus exemplos não cobrem todos os usos da linguagem e que, por essa razão, não fica demonstrado que a função operacional é pressuposta todas as vezes em que alguém usa a linguagem de forma eficiente. Efetivamente, alguém poderia querer dar outros exemplos, para mostrar que há casos em que a linguagem não cumpre uma função operacional. Poder-se-ia pensar que os performativos de Austin fornecem exemplos assim.

Por exemplo, quando alguém se casa e pronuncia a palavra “aceito”, poder-se-ia dizer que essa pessoa realiza uma ação pelo simples fato de pronunciar a palavra, e não importa em nada se a palavra é usada como código ou não. Bem, para começar, eu diria que alguém só poderia dar um exemplo assim se fizesse uma análise muito superficial da situação. No meu modo de ver, está claro que a palavra tem uma função operacional. É verdade que algumas pessoas

precisam ouvir a palavra para que os noivos sejam considerados casados, em particular, a pessoa investida da autoridade para casá-los deve ouvi-la. Mas não é só o som que conta. A informação fornecida pela palavra é essencial. Essa informação precisa ser computada pela autoridade presente para que os noivos sejam declarados casados. Se em vez de "aceito", um dos noivos dissesse a palavra "não", o resultado da cerimônia não seria o de tornar os noivos casados. Na minha opinião, os resultados diferentes provocados pela palavra "aceito" e pela palavra "não" devem ser atribuídos ao fato das palavras veicularem informações diferentes.

Um outro exemplo mais interessante é o do uso de fórmulas mágicas. É sabido que muitas pessoas usaram, e algumas ainda usam, fórmulas ou palavras mágicas. Essas pessoas acreditavam, ou acreditam, que a simples pronúncia dessas fórmulas e palavras causa mudanças na realidade. Por exemplo, na estória de Ali Babá e os quarenta ladrões, "Abre-te, Sésamo!" é a fórmula mágica que abre a porta da caverna dos ladrões. É claro que a estória é fictícia, mas vamos usá-la aqui para ilustrar uma questão real, a questão do uso mágico da linguagem. A questão é: quando os ladrões usavam essa fórmula eles a usavam como código? A fórmula veiculava alguma informação? A resposta é "sim". Desde que a porta se abria, a fórmula era usada como um código, no mesmo sentido em que uma chave é usada como um código. O que é inusitado na situação é que a frase não é decodificada por um sistema na mente de um ser humano. A decodificação é feita pela porta. De fato, como foi demonstrado pelo irmão de Ali Babá, se outra fórmula fosse pronunciada, a porta não se abria, o que indica que o sistema de decodificação da porta era tal que só podia ser acionado pela informação fornecida pela fórmula "Abre-te, Sésamo!". No fundo, era uma porta como todas as outras (pelo menos como todas as outras deveriam ser): só abria com a sua chave. Mas sejamos mais realistas. O que aconteceria se alguém tentasse de fato abrir uma porta pronunciando uma fórmula ou palavra? Isso seria possível? Certamente. Uma porta acionada eletronicamente pode efetivamente ser programada para abrir quando alguém pronuncia uma palavra. Se alguém pronunciar a palavra errada, ela não se abrirá. Nessa situação, é mais do que claro que a palavra é um código e que a porta tem um sistema que processa o código. Mas examinemos um caso mais difícil. Alguém fica parado diante de uma porta comum e pronuncia a palavra "abracadabra" na esperança de que a palavra abra a

porta. A porta não se abre. Como se explica esse caso? Creio que nesse caso, embora o falante tenha a intenção de usar a palavra como código, ela não fornece a informação que ele espera que ela forneça. Ela poderia fornecer a informação se a porta tivesse um sistema como o da porta eletrônica. O problema é que ela não tem, e isso faz com que a palavra não surta nenhum efeito com ela. O código que surtiria efeito seria provavelmente uma certa chave girando no sentido horário. Dessa forma, pode-se concluir que a palavra, naquele uso específico, não desempenha uma função operacional. Isso refuta a minha tese de que a função operacional é a função básica da linguagem? Claro que não. Minha tese é de que a função operacional é pressuposta em todo uso *eficiente* da linguagem. Nos casos ineficientes, a linguagem não desempenha uma função operacional porque simplesmente não desempenha função nenhuma.

Mesmo assim, é bom lembrar que um mesmo dado pode fornecer informações diferentes para diferentes propósitos, e pode até mesmo carecer de informação para um propósito e fornecê-la quando o propósito é diferente. Nesse sentido, mesmo quando a palavra "abracadabra" é usada do modo descrito acima, ela pode ser informativa. É certo que ela não fornece informação para abrir a porta, mas fornece informação para alguém que ouça ela ser usada como palavra mágica. Com base nessa informação, o ouvinte pode dizer: "fulano acha que dizendo 'abracadabra' vai fazer a porta se abrir. Ele está louco!". Dessa forma, quando considerada em vista de outro propósito, a palavra ainda pode ser considerada informativa.

É preciso lembrar, no entanto, que o que está em questão é o uso de palavras e fórmulas para realizar efeitos mágicos. E, para tal questão, uma vez que foram analisados vários casos de palavras pronunciadas para abrir uma porta, já temos uma explicação. Se uma expressão é dita na qualidade de expressão mágica e produz o efeito esperado, então algum sistema decodificou a palavra e, nesse caso, a palavra tem uma função operacional. *Per contra*, se a expressão não produz o efeito esperado, é porque ela não fornecia informação para aquele propósito e, nesse caso, a palavra não cumpre função alguma.

Um último exemplo (pelo menos o último que pude imaginar) que poderia ser usado para tentar mostrar que a linguagem pode ser usada eficientemente sem desempenhar uma função operacional é o exemplo do monólogo. Alguém poderia achar estranho que o emissor do código fosse ao

mesmo tempo responsável por sua decodificação. Tenho de dizer, porém, que o exemplo não atinge em nada minha tese. Não há qualquer impedimento para que a mesma pessoa fale consigo mesmo e entenda o que falou. Se a pessoa entende, ela decodifica. Poder-se-ia alegar que o monólogo não gera nenhuma mudança de estado na mente da pessoa e isso demonstraria que nenhum processamento foi feito. Mas não é verdade que não houve mudança de estado. Parece sensato acreditar que num monólogo, não só há mudanças emocionais no falante-ouvinte, mas há até mesmo mudanças cognitivas. Certas lembranças são ativadas, novos conteúdos são inferidos etc.

Acredito que o exame desses exemplos mostra que de fato não há possibilidade de uma expressão da linguagem ser usada de modo eficiente, seja qual for o propósito para o qual ela é designada, se não fornecer informação para um sistema de decodificação apropriado. Com os nomes próprios acontece a mesma coisa. Um nome próprio só realiza uma função se ele é usado como código. Com efeito, podemos verificar isso facilmente quando o nome é usado como vocativo.

Vejamos um exemplo. Eu estou andando despreocupado e de repente ouço alguém gritar meu nome atrás de mim. Em tal caso, o que eu faço em geral é me virar e procurar quem está me chamando. Por que eu faço isso? A resposta mais sensata parece ser a de que eu faço isso porque, ao ouvir meu nome, eu penso: “alguém está me chamando”, ou algo assim. O fato de eu ouvir o meu nome causa uma mudança de estado em mim. Isso indica que o nome me fornece uma informação. O resultado da decodificação dessa informação é o ato de me virar e procurar. Também em outros casos de nomes usados como vocativos, verifica-se que sempre há um efeito causado pelo nome. Em um lugar com muitas pessoas, um nome próprio pode ser usado como vocativo para selecionar o interlocutor do falante. Na sala de aula, por exemplo, eu posso perguntar: “Zezinho, qual o sentido da vida?”, e ao usar o nome “Zezinho” eu seleciono o aluno que terá a árdua tarefa de responder a minha pergunta. O resultado do meu uso do nome é que o Zezinho vai ter que se esforçar para responder minha pergunta. Por que o Zezinho tem que fazer todo o esforço e não algum outro aluno? Porque ao ouvir o seu nome, o Zezinho é informado de que, dentre todos os seus colegas, ele é o escolhido para responder minha pergunta. Algumas vezes, porém, o efeito de usar um nome como vocativo é mais sutil. Se eu digo “Tchau!”

para uma garota isso tem um certo efeito. Se eu digo: “Tchau, Frederica!” isso tem um outro efeito. Aparentemente, uma expressão de despedida, ou uma saudação, ou um pedido, ou um impropério fica mais pessoal quando está ligado ao nome do interlocutor. Nesse caso, o nome contribui para o efeito do todo, e essa contribuição pode ser vista como um acréscimo de informação. Ou seja, nesse caso também o nome próprio é informativo. Em resumo, tudo indica que, quando é usado como vocativo, um nome próprio sempre fornece alguma informação para o ouvinte. Ele sempre exerce uma função operacional em tais casos.

Quando um nome próprio é usado numa sentença interrogativa, como em “quem foi Napoleão?”, ou em ordens, como em “vá chamar a Dalila!”, também parece fácil admitir que ele precisa fornecer alguma informação para nossos sistemas de decodificação, caso contrário o interlocutor não poderia dar a resposta nem cumprir a ordem. O próprio Kripke admite que quando alguém lhe pergunta a quem ele refere com o nome “Napoleão”, ele responde algo como “Napoleão foi imperador da França na primeira parte do século XIX; ao final, ele foi derrotado em Waterloo” (KRIPKE, 1972: 28). Isso parece indicar que, na pergunta, o nome “Napoleão” funciona como um código que, ao ser decodificado, leva o interlocutor a dar uma certa resposta. Se a pergunta fosse “quem foi Robespierre?”, a resposta seria diferente. Se fosse “quem foi Maria Antonieta?”, teríamos ainda outra resposta. O fato de a resposta mudar de acordo com o nome que aparece na pergunta parece indicar que cada nome fornece uma informação diferente. Mesmo quando a resposta a tais perguntas é “não sei”, ainda assim o nome em questão precisou passar pelo sistema de decodificação do interlocutor. O sistema buscou certas informações na memória do interlocutor e, não as achando, produziu a resposta “não sei”. Só pelo fato do sistema executar essas ações, já se pode dizer que o nome desempenhou uma função operacional. Ele disparou certas operações no sistema.

Argumentos semelhantes podem ser aduzidos paro o caso de ordens como as do tipo “vá chamar a Dalila!”. O interlocutor só pode executar uma ordem desse tipo se ele reconhecer o nome, verificar se tem alguma informação associada ao nome e utilizar essa informação para localizar e efetivamente chamar a Dalila. E, semelhantemente ao caso anterior, pode acontecer de ele não conhecer a Dalila. Em todo caso, para poder verificar que não possui nenhuma informação

sobre tal pessoa, ele precisa em primeiro lugar processar o nome. Esse processamento é o que, em tais situações, leva o interlocutor a perguntar: “quem é Dalila?”. Dessa forma, fica claro que o nome sempre desempenha uma função operacional.

Já quando um nome próprio é usado em uma sentença declarativa [5] como sujeito, como objeto ou em qualquer outra função que não seja a de vocativo, pode não ser tão fácil ver de que forma o nome desempenha uma função operacional. Isso acontece porque o efeito do nome no interlocutor muitas vezes não é visível. Com efeito, durante a leitura de um texto eu posso deparar com uma porção de nomes próprios e continuar aparentemente impassível diante deles. Acredito, por exemplo, que minha reação ao ler o nome “Wittgenstein” pela primeira vez não foi tão notável assim. Apesar disso, algo aconteceu em mim quando li esse nome (esse pode parecer o relato de uma epifania, mas de fato estou falando de algo bem banal). O que aconteceu em mim em tal ocasião? O que aconteceu em mim quando li no jornal que Chuck Berry viria fazer uma apresentação em Fortaleza? O que aconteceu quando ouvi os nomes “Joaquim” e “Manuel” na piada de que falei anteriormente [6]? Admito que o efeito causado pelo nome pode não ser tão óbvio nesses casos, mas também não é impossível de ser identificado. Acredito que para identificá-lo adequadamente, devemos antes pensar minimamente sobre as operações que nossos sistemas internos de decodificação realizam assim que são ativados por um proferimento.

Minha hipótese é de que a primeira coisa que eles têm de fazer é realizar um tipo de *check in* de dados. Nesse *check in*, presumo que o sistema terá que fazer pelo menos três coisas: 1. reconhecer as palavras; 2. identificar a função sintática das palavras e 3. identificar o tipo de proferimento que está sendo analisado. Esse tipo de vistoria preliminar dos dados tem que ser feita antes que outros procedimentos sejam acionados. Alegoricamente falando, é como se o proferimento fosse um pacote cheio de informação, só que em cima do pacote há uma informação adicional dizendo como a informação de dentro do pacote deve ser distribuída e a quem cada parte da informação está endereçada. Antes de

[5] Deve-se notar que uma sentença é considerada declarativa em razão de sua estrutura sintática. A questão de se a sentença declara algo sobre a realidade não importa. Dessa forma, sentenças declarativas podem tanto ser descritivas como ficcionais.

[6] Note que, na piada, o riso não estava relacionado aos nomes (embora seja preciso confessar que os nomes tinham relação com o contexto da piada).

repassar o conteúdo do pacote para quem de direito, o sistema deve ler as instruções na frente do pacote. Dito de forma direta, quando o sistema recebe um código, a primeira coisa que ele faz é checar as informações preliminares que orientarão o restante do processamento. Isso é o que acontece no *check in* de dados. Depois dessa primeira checagem, as outras informações contidas no código são passadas a sistemas especializados, e aí tem início a segunda fase do processamento. É nessa fase posterior que será realizada a tarefa para a qual o código efetivamente é emitido. O *check in* de dados serve essencialmente para selecionar e classificar a informação e os parâmetros que serão processados na segunda fase.

O item do *check in* que tem maior importância na discussão sobre nomes próprios é o primeiro, o que trata do reconhecimento das palavras. Ele também é o mais óbvio. Basta lembrar um exemplo anterior – um texto em russo não pode ser decodificado por um leitor que não sabe russo. No entanto, se um leitor que não sabe russo por acaso deparasse com um texto em russo, ele saberia que não pode ler aquele texto. Ele saberia disso porque seu sistema de decodificação de linguagem tentaria processar o texto e não teria sucesso. Imagino que o procedimento que o sistema usaria em tal caso seria o seguinte: palavras do texto seriam selecionadas e comparadas com o vocabulário do leitor; uma vez que as palavras não fossem reconhecidas, o sistema produziria um sinal indicando que ele não está apto a decodificar o texto. É assim que o referido leitor percebe sua incapacidade de ler o texto.

Na minha opinião, esse procedimento de reconhecimento de palavras deve acontecer sempre. Até nomes próprios precisam passar por essa etapa de reconhecimento. Não resta dúvida de que ele é essencial quando o nome aparece em perguntas como “quem é Napoleão?” e em ordens como “vá chamar a Dalila!”. Em um dos casos, o nome precisa ser reconhecido para que uma resposta seja dada; no outro caso, o nome precisa ser reconhecido para que uma pessoa seja chamada. Mas também em sentenças declarativas o reconhecimento do nome deve ser feito obrigatoriamente. Digamos, por exemplo, que eu leio o seguinte enunciado: “Wittgenstein gostava de musicais”. Quando eu leio esse enunciado, meu sistema interno de decodificação de linguagem imediatamente confere as palavras do enunciado e as relaciona com informações que se encontram registradas em minha memória, isso, é claro, se as palavras forem reconhecidas.

Com a palavra “Wittgenstein”, não é diferente. Ela também precisa passar pelo processo de reconhecimento. Se quisermos lançar mão de uma analogia, podemos dizer que eu guardo minhas informações sobre Wittgenstein em um certo arquivo, e que o nome “Wittgenstein” abre esse arquivo. Mas o que aconteceu quando ouvi o nome “Wittgenstein” pela primeira vez? Digamos que tenha sido na sentença “Wittgenstein era austríaco”. Posso apostar que naquele dia apliquei o mesmo procedimento de reconhecimento ao nome. A diferença é que quando meu sistema de decodificação foi verificar se já existia alguma informação que lhe pudesse ser associada, não havia nenhuma. O que eu fiz então? Simples, abri um novo arquivo para colocar informações sobre Wittgenstein e inaugurei esse arquivo com a informação que recebi por meio da sentença “Wittgenstein era austríaco”. Não fiz nenhum trejeito peculiar quando fiz isso, tudo aconteceu internamente. Ou seja, o efeito que o nome provocou em mim não foi visível. No entanto, o efeito existiu e envolveu várias operações, entre as quais a operação de reconhecimento do nome. O fato de que essas operações foram deflagradas pelo nome nos autoriza a afirmar que o nome desempenhou uma função operacional. Nesse sentido, ele foi informativo.

Aqui é preciso ter muita atenção. A ideia de que um nome n contém informação não equivale à ideia de que n é sinônimo de alguma outra expressão da linguagem ou de que n diz algo sobre algum particular. Um nome contém informação simplesmente porque causa uma mudança de estado no meu sistema de decodificação de linguagem, ele desencadeia certas operações nesse sistema. É nesse sentido, por exemplo, que os nomes “Manuel” e “Joaquim” são informativos. Na piada, eles não são usados para referir ninguém. A situação protagonizada por Manuel e Joaquim é fictícia. Dessa forma, os nomes não podem dar nenhuma informação sobre um particular. Na verdade, a informação que eles veiculam é antes de tudo uma informação para a operação de um sistema de decodificação. É muito importante fazer essa diferença entre *informação sobre* e *informação para*. A informação veiculada pelos nomes próprios é em primeiro lugar informação para. Ela serve para acionar certos processos mentais de computação. Entretanto, uma vez que esses processos são acionados, os nomes passam a ser associados a outras informações, e essas informações possibilitam o seu uso em novas construções linguísticas, algumas das quais podem até servir para descrever fatos do mundo (e.g., “você está igual ao Joaquim da piada”). Mas

isso também vale para nomes de coisas reais. Apesar do nome em geral não conter em si mesmo informação descritiva sobre a coisa que ele nomeia, uma vez que seja processado, ele sempre é associado a informações que podem ajudar a caracterizar a tal coisa. Essa ideia está relacionada com o pensamento que Searle expressa quando afirma que nomes próprios "não funcionam como descrições, mas como pegadores nos quais nós penduramos descrições" (SEARLE, 1958: 172). Essa relação deve ficar mais clara no próximo capítulo, quando for analisar de que modo a função operacional de um nome próprio influencia na sua função referencial. De todo modo, deve-se observar que nomes próprios só podem ser associados a informações-sobre porque podem ser processados, e isso, por sua vez, só acontece porque eles sempre fornecem a nossos sistemas de decodificação informações-para. Nesse sentido, pode-se dizer que as palavras da língua russa fornecem informação para o leitor que não sabe ler russo. Mas elas fazem isso apenas durante a fase do processamento que chamei de "*check in* de dados". Quando a informação consegue passar dessa fase, outras operações têm início, e são essas operações que o leitor que não sabe russo não consegue levar a efeito. Essas operações constituem a fase do processamento especializado. Na próxima seção deste capítulo, vou tratar mais especificamente de algumas operações que ocorrem nessa segunda fase.

Não acho necessário falar muito sobre os processos correspondentes aos itens 2 e 3 do *check in* de dados. Só gostaria de fazer algumas observações rápidas a respeito deles. Uma observação que serve para os dois é a de que eles, a exemplo do item 1, também servem para preparar o caminho para a fase do processamento especializado. Eles também lidam com informações preliminares. As outras observações são específicas e por isso as farei separadamente.

A identificação da função sintática das palavras, que corresponde ao item 2 do *check in* de dados, é uma parte essencial do processamento linguístico. Com efeito, é graças a esse tipo de informação que somos capazes de reconhecer a diferença entre "Tom ama Maria" e "Maria ama Tom", por exemplo. É razoável pensar que, já no *check in* de dados, o sistema deve ser capaz de ler essa informação sintática, pois essa leitura é que vai determinar quais operações vão ser aplicadas a quais palavras na segunda fase do processamento. Além disso, a leitura da informação sintática deve ser realizada de forma independente do processo de reconhecimento das palavras. De fato, Chomsky ilustrou esse ponto

com sua famosa frase “Twas brillig and the slithy toves did gyre and gimble”, que, apesar de não ser inteligível, ainda fornece um tipo de informação sintática para o leitor. Da mesma forma, Lewis Carroll explorou essa nossa capacidade de processar informação sintática independentemente de informação semântica em seu poema Pargarávio (“Jabberwocky”) [7].

Outrossim, o processo correspondente ao item 3 é indispensável para o processamento da informação linguística. É esse processo que verifica o tipo de proferimento que vai ser decodificado, ou seja, ele identifica a finalidade do proferimento. Para isso, o processo leva em consideração tanto informações sintáticas como informações fornecidas pelo contexto do proferimento. Dependendo de qual seja a finalidade do proferimento, ele será encaminhado para um subsistema especializado ou outro. Esse subsistema cuidará de atender aquela finalidade. Se o proferimento é uma sentença declarativa, o sistema deve encaminhar a informação para seus subsistemas que tratam dessa área. Se o proferimento é uma pergunta, o sistema deve encaminhar a informação para seus subsistemas que respondem perguntas. Se o proferimento é uma ordem que pede uma resposta motora, o sistema deve encaminhar a informação para seus subsistemas que processam sentenças imperativas e acionam sistemas motores etc. Em suma, quando nossos sistemas internos de decodificação recebem um proferimento, eles precisam decidir o que fazer com ele antes mesmo de processá-lo de uma forma especializada. Essa decisão ocorre no *check in* de dados, assim que o processo de identificação do tipo do proferimento é completado.

É muito importante salientar que o funcionamento desses processos é muito mais complexo e muito mais integrado do que as breves observações que fiz podem dar a entender. Por exemplo, o processo de reconhecimento de palavras deve recorrer assiduamente ao processo de análise da informação sintática e ao processo de análise da informação contextual. Reconhecer uma palavra não significa simplesmente reconhecer um sinal gráfico ou sonoro. O uso que se faz do sinal em dada ocasião também é fundamental, e a identificação desse uso depende de informação sintática e contextual. É graças à informação sintática que,

[7] Pargarávio é um poema que aparece em *Alice no País dos Espelhos* e cuja autoria é atribuída ao personagem Humpty Dumpty. Quase todo o poema é composto de palavras inventadas. Para se ter uma ideia, o primeiro verso do poema diz o seguinte: “Solumbrava, e os lubriciosos touvos em vertigiros persondavam as verdentes”. O interessante é que mesmo sendo indecifrável para quem não está a par do vocabulário de Humpty Dumpty, o poema parece fornecer um certo tipo de informação. Essa informação é a informação sintática.

por exemplo, podemos reconhecer dois usos diferentes da palavra “rosa” na sentença “a rosa é rosa”. Já na sentença “a hipotenusa brincava alegremente no parque”, embora a palavra “hipotenusa” possa ser reconhecida, seu uso não é tão fácil de reconhecer, e isso compromete a decodificação da sentença. Isso mostra que o reconhecimento do uso de uma palavra que aparece em um determinado contexto depende do reconhecimento das outras palavras que aparecem com ela no mesmo contexto.

Essas são, em resumo, as observações que queria fazer sobre a fase do *check in* de dados. Nessa fase, o que mais importa para a compreensão do processamento de nomes próprios é o processo de reconhecimento das palavras. Durante esse processo, até um nome próprio precisa ser reconhecido e esse reconhecimento depende de certas informações preliminares fornecidas pelo nome. Só isso já mostra que nomes são informativos, o que equivale a dizer que eles cumprem uma função operacional. Além disso, essa deve ser a função básica do nome, pois se ela não for exercida, isso significa que o proferimento do qual o nome faz parte não foi decodificado, e, como sabemos, um proferimento que não é decodificado é um proferimento que não cumpre sua finalidade, seja ela qual for.

É possível, porém, particularizar mais o problema e falar das funções específicas dos nomes próprios. Com relação à função de vocativo, por exemplo, vimos que o nome só é capaz de desempenhar tal função se antes de tudo ele desempenhar uma função operacional. Se a função do nome é nomear um personagem ficcional, a mesma coisa. Em relação à função referencial, não é diferente. Era essa função que o nome “Napoleão” e o nome “Dalila” pretendiam desempenhar em exemplos anteriores. Como acredito deve ter ficado claro, esses nomes só cumpriam sua função referencial porque antes de tudo desempenhavam uma função de código, ou seja, uma função operacional.

Mas, apesar dos argumentos já apresentados, é possível esclarecer ainda mais a dependência que a função referencial tem em relação à função operacional dos nomes próprios se adquirirmos mais algumas intuições sobre o processamento dos nomes próprios na segunda fase do processamento linguístico feito por nossos sistemas internos de decodificação, vale dizer, na fase do processamento especializado. Tento oferecer essas intuições na seção seguinte, na qual descrevo um programa com a função específica de responder a pergunta

"quem é X?", sendo X um nome próprio qualquer. O programa é descrito como um programa de computador, mas é proposto como um modelo do processamento que a mente faz quando um agente de linguagem humano responde a mesma pergunta.

6.4. ProgX: um modelo para o processamento mental de nomes próprios

Digamos que quiséssemos programar um computador para responder a pergunta "quem é X?", sendo X um nome próprio qualquer. Chamemos esse programa de ProgX. O que ProgX teria que fazer para responder a tal pergunta? Uma possibilidade seria começarmos com um banco de dados repleto de informações proposicionais que pudessem ser associadas a X e que fossem acessadas pelo programa assim que ele recebesse a pergunta "quem é X?". Nesse caso, ProgX seria apenas um programa de busca. Perguntado quem é X, o programa buscaria no banco de dados informações sobre X e a partir daí daria uma resposta. Mais interessante, porém, seria se o banco de dados fosse gerado por uma sub-rotina de ProgX. Chamemos essa sub-rotina de Sub1. Uma vez gerado o banco de dados, uma outra sub-rotina Sub2 teria a função de usar o nome "X" que ocorre na pergunta como chave para localizar as informações associadas a "X", utilizando em seguida essas informações para compor uma resposta. Em linhas gerais, é isso que ProgX teria que fazer para responder a pergunta. Doravante descreverei mais detalhadamente os processos que Sub1 e Sub2 teriam que implementar para executar as tarefas que lhes concernem.

Para que Sub1 execute a tarefa que lhe concerne, ela deverá realizar pelo menos duas operações principais: 1. ponderar informação e 2. organizar informação, sendo que para realizar essa segunda operação, Sub1 terá que realizar uma série de outras operações, entre elas: ler, escrever e inferir informação, reconhecer palavras e solucionar problemas de ambiguidade e inconsistência.

Vou assumir que os dados que Sub1 recebe são enunciados. A rotina recebe um enunciado quando um operador efetivamente digita um enunciado e o insere em Sub1. Para efeito de simplificação, vou assumir também que todos os operadores de ProgX são competentes e bem intencionados e, por isso, só

introduzem no programa enunciados gramaticalmente corretos, embora possam introduzir enunciados dedutíveis de enunciados já registrados e até mesmo enunciados inconsistentes com enunciados já registrados.

Outrossim, é importante esclarecer que, embora ProgX seja idealizado para modelar uma situação de uso de nomes próprios, os enunciados inseridos em Sub1 não precisam ter ocorrências de nomes próprios. A informação fornecida por enunciados sem ocorrência de nomes próprios pode ser útil para que novos enunciados contendo nomes próprios sejam inferidos. Por exemplo, um enunciado como “professores em geral ganham pouco” pode ser combinado com o enunciado “Pardal é professor”, dando assim condições para que o programa infira que Pardal provavelmente ganha pouco.

Ao receber um enunciado, a primeira coisa que Sub1 terá que fazer é atribuir um certo peso à informação fornecida por esse enunciado. Para tanto, Sub1 deve recorrer a uma tabela que dispõe os operadores do programa em categorias e relaciona essas categorias com certos valores numéricos. Se Sub1 receber uma informação através de um enunciado inserido por um operador da categoria n, o peso que Sub1 atribuirá à informação será n. A utilidade desses pesos que Sub1 atribuirá às informações será evidenciada mais adiante.

A operação de organizar informação é bem mais complexa. O que Sub1 deve fazer é ler a informação fornecida pelos enunciados que ela recebe, armazenar essa informação no banco de dados e construir uma espécie de índice remissivo para essas informações ao qual chamarei de “índice de endereçamento”. No banco de dados, cada informação tem um endereço e, no índice de endereçamento, esses endereços são anexados às palavras advindas dos enunciados previamente processados. Assim, no índice de endereçamento, Sub1 escreverá itens do tipo [8]:

JOÃO (A1, A7, B3, C10, J5)
AMAR (E1, B4, C10, E20, E345,..., G31)

[8] É bom notar que esse é só um simbolismo que me ajuda a tornar mais clara a descrição de ProgX. Todavia, a linguagem que o programa deve utilizar não é algo que quero estabelecer aqui. O próprio sistema alfanumérico que uso para os endereços é apenas uma sugestão. A ideia é que as letras poderiam servir para indicar setores no banco de dados e os números segmentos dentro desses setores (metaforicamente falando, o sistema indicaria as ruas e as casas dentro do banco de dados).

PROFESSOR (A7)
MARIA (A90, E1, H13)

Como se pode ver, cada um desses itens relaciona uma palavra com um ou mais endereços. O primeiro item, por exemplo, relaciona a palavra "João" com os endereços A1, A7, B3, C10 e J5 o que significa que a palavra "João" foi identificada nos enunciados que forneceram as informações registradas nos endereços A1, A7, B3, C10 e J5. Quando o nome aparece em um novo enunciado, a lista de endereços que o segue é atualizada e passa a incluir o endereço da informação fornecida pelo novo enunciado.

É importante chamar a atenção para o fato de que alguns itens podem conter palavras homônimas, ou seja, um determinado item pode apresentar a mesma palavra que aparece em outro item. O que vai distinguir esses itens é a lista de endereços anexada a cada palavra homônima. Por exemplo, JOÃO (A1, A7, B3, C10, J5) e JOÃO (D20, J2, S5) são itens diferentes. Esses itens que contêm palavras homônimas serão chamados, a propósito, de "itens homônimos". No banco de dados, a representação das informações relacionadas a um item homônimo também deve ser diferenciada da representação das informações relacionadas aos outros itens homônimos. Assim, por exemplo, digamos que Sub1 recebe o enunciado S_1= "João é padeiro" e o relacione com um dos itens acima, e que recebe o enunciado S_2= "João é neurocirurgião" e o relacione com o outro item. Se Sub1 representar a informação fornecida por S_1 como, digamos, *João é padeiro*, a rotina deve representar a informação fornecida por S_2 de uma forma tal que um João possa ser diferenciado do outro, por exemplo, ela pode registrar que *João$_2$ é neurocirurgião* [9].

Outra coisa importante sobre o índice de endereçamento é que alguns de seus itens podem se encontrar *encadeados* (a importância disso se tornará evidente logo adiante). Direi que os itens

[9] Nesse caso também, o simbolismo é só um recurso usado para facilitar minha exposição. Não pretendo me aprofundar na questão de como a informação é representada no banco de dados. É certo que toda informação precisa ser representada por meio de um código, mas não cabe a mim eleger a linguagem que efetivamente deve ser usada para codificar a informação registrada por Sub1.

I_1 e I_2 estão encadeados (em símbolos: $I_1 – I_2$) se e somente se eles têm pelo menos um endereço em comum

E que os itens

I_1, ..., I_n estão encadeados se e somente se, para algum item I_j, onde $0<j\leq n$, os itens I_1, ..., I_{j-1}, I_{j+1}, ..., I_n estão encadeados e I_j está encadeado com algum desses itens.

Direi que itens que não estão encadeados estão “desencadeados”. Chamarei de “encadeamento” um conjunto de itens encadeados e de “elo” cada endereço comum entre itens encadeados. Finalmente, chamarei um conjunto de elos de “itinerário”.

Fica evidente que um itinerário dá os endereços das informações relacionadas a um encadeamento e que para cada encadeamento há um itinerário correspondente.

Alguns exemplos podem ajudar a esclarecer essas noções. Se chamarmos os itens listados acima respectivamente de I_a, I_b, I_c e I_d, teremos as seguintes verdades:

$I_a – I_b$	({C10} é o itinerário correspondente)
$I_a – I_c$	({A7} é o itinerário correspondente)
$I_b – I_d$	({E1} é o itinerário correspondente)
$I_a – I_b – I_d$	({C10, E1} é o itinerário correspondente)
$I_a – I_b – I_c$	({C10, A7} é o itinerário correspondente)
$I_a – I_b – I_c – I_d$	({C10, E1, A7} é o itinerário correspondente)

Mas não será verdade que:

$I_a – I_d$

$I_b – I_c$

$I_c – I_d$

$I_a – I_c – I_d$

$I_b – I_c – I_d$

Note que $I_a - I_b - I_c - I_d$, mas não é verdade, por exemplo, que $I_b - I_c - I_d$, o que significa que mesmo quando um conjunto de itens é um encadeamento não é necessário que todas as suas partes sejam encadeamentos. Por outro lado, enquanto é falso que $I_b - I_c - I_d$, é verdade que $I_b - I_d$, o que mostra que mesmo quando um conjunto de itens não constitui um encadeamento algumas de suas partes podem ser encadeamentos. Note também que a relação de encadeamento é reflexiva, isto é, $I_n - I_n$, para todo I_n. E é também simétrica, isto é, se $I_n - I_m$, então $I_m - I_n$. Em contrapartida, ela não é transitiva, isto é, mesmo quando $I_m - I_n$ e $I_n - I_o$, pode não ser o caso que $I_m - I_o$. Ademais, é possível mostrar que, se dois encadeamentos têm um item em comum, então a união desses encadeamentos é um encadeamento. Uma prova seria a seguinte:

A e B são encadeamentos e $A \cap B = I_{k1}$ (hip.)
$A \cup \{I_{k1}\}$ é um encadeamento
Existe um item I_{k2} de $B-\{I_{k1}\}$ que está encadeado com I_{k1}, logo
$A \cup \{I_{k1}, I_{k2}\}$ é um encadeamento
Existe um item I_{k3} de $B-\{I_{k1}, I_{k2}\}$ que está encadeado ou com I_{k1} ou com I_{k2}, logo
$A \cup \{I_{k1}, I_{k2}, I_{k3}\}$ é um encadeamento
E assim por diante, até a conclusão de que
$A \cup B$ é um encadeamento

Dado um item I_k qualquer, há um encadeamento que contém todos os encadeamentos que incluem I_k. Esse encadeamento será formado por todos os itens do índice de endereçamento que se relacionam com I_k direta ou indiretamente. Vou chamar tal conjunto de "encadeamento máximo de I_k". Um procedimento para achar o encadeamento máximo de I_k é o seguinte:

Estágio 1: geram-se todos os encadeamentos binários que incluem I_k, ou seja, geram-se todos os encadeamentos do tipo $\{I_k, I_{x1}\}$.

Estágio 2: Para cada I_{x1}, geram-se todos os encadeamentos binários que incluem I_{x1}, menos os que são repetições de encadeamentos gerados no estágio anterior, ou seja, geram-se todos os encadeamentos do tipo $\{I_{x1}, I_{x2}\}$, tal

que $\{I_{x1}, I_{x2}\} \neq \{I_k, I_{x1}\}$. Em seguida, toma-se o primeiro encadeamento gerado no estágio 1 e opera-se a sua união com todos os encadeamentos binários gerados em 2 que tem um item em comum com ele. Faz-se operação similar com todos os outros encadeamentos gerados no estágio 1. Assim, geram-se todos os encadeamentos ternários que incluem I_k, ou seja, geram-se todos os encadeamentos do tipo $\{I_k, I_{x1}, I_{x2}\}$.

No estágio n, para cada I_{xn-1}, geram-se todos os encadeamentos binários que incluem I_{xn-1}, menos os que são repetições de encadeamentos gerados no estágio anterior, ou seja, geram-se todos os encadeamentos do tipo $\{I_{xn-1}, I_{xn}\}$, tal que $\{I_{xn-1}, I_{xn}\} \neq \{I_{xn-2}, I_{xn-1}\}$. Em seguida, toma-se o primeiro encadeamento de cardinalidade n gerado no estágio n-1 e opera-se a sua união com todos os encadeamentos binários gerados em n que tem um item em comum com ele. Faz-se operação similar com todos os outros encadeamentos gerados no estágio n-1. Assim, geram-se todos os encadeamentos de cardinalidade n+1 que incluem I_k, ou seja, geram-se todos os encadeamentos do tipo $\{I_k, I_{x1},..., I_{xn}\}$.

A partir de um certo estágio, as uniões dos encadeamentos começam a convergir e, finalmente, no último estágio, gera-se o encadeamento máximo de I_k. Como o encadeamento máximo de I_k não pode ter mais elementos que o conjunto de todos os itens (que obviamente é finito), é necessário que o número de estágios do processo seja menor do que o número de elementos do conjunto dos itens.

Uma vez dadas essas explicações iniciais, torna-se possível descrever o modo como Sub1 executa o processo de organizar informação. O processo vai ser executado em sete passos:

1. Sub1 recebe um enunciado S e procura as palavras componentes de S no índice de endereçamento [10]. Se uma palavra não é localizada em nenhum item do índice, ela é classificada como não localizada. Em seguida, para cada item I_k no qual Sub1 encontra uma palavra w de S, a rotina pergunta ao operador: “Você usa a palavra “w” no mesmo sentido em que ela é

[10] Algumas palavras serão dispensadas desse processo de verificação. Essas palavras são os artigos, as preposições, as conjunções, os pronomes, alguns advérbios (e.g. “não”, “sempre”, “nunca”) e o verbo “ser”. Em todo caso, ProgX deve poder lidar com essas palavras quando precisar inferir informações novas a partir daquelas registradas no banco de dados. Como veremos, o programa é dotado de um provador que realiza todas as inferências que ele solicita.

usada nos enunciados S_1,..., S_n, sim ou não?", onde S_1,..., S_n expressam as informações relacionadas a I_k [11]. Se a resposta do operador é "sim", Sub1 seleciona I_k e a palavra w é classificada como localizada. Senão, Sub1 não seleciona I_k. Se a resposta do operador é "não" para todo item que contém a palavra w, w é classificada como não localizada.

2. Se todas as palavras de S são localizadas no passo 1, uma sub-rotina R1 é chamada para fazer dois testes com S. TESTE1 (teste de dedutibilidade): R1 verifica se S é dedutível dos enunciados previamente processados por Sub1. TESTE2 (teste de consistência): R1 verifica se ¬S é dedutível dos enunciados previamente processados por Sub1. Se S tem alguma palavra w classificada como não localizada, R1 é chamada para realizar os mesmos testes com S', sendo S' um enunciado igual a S, a não ser por apresentar o *string* 'w*' nos lugares em que S apresenta o *string* 'w'.
3. Sub1 utiliza os resultados dos testes realizados por R1 da seguinte forma: Se o TESTE1 der positivo, Sub1 é encerrada (o programa não precisa registrar uma informação se ela pode ser inferida das informações já registradas). Se o TESTE2 der positivo, Sub1 chama uma sub-rotina R2 que terá a função de desfazer a inconsistência. Para tanto, R2 irá optar entre (I) apagar uma das informações já registradas envolvidas na inconsistência, apagando juntamente o endereço dessa informação em todos os itens em que ele aparece, e (II) desconsiderar a informação fornecida por S. No caso de (I), Sub1 passa para 4. No caso de (II), Sub1 é encerrada. Se os dois testes de R1 dão negativo, então Sub1 vai para 4.
4. Sub1 registra a informação fornecida por S com seu respectivo peso em um endereço Xn do banco de dados e acrescenta o endereço Xn em todos os itens selecionados no passo 1.
5. As palavras não localizadas no passo 1 são escritas no índice de endereçamento e junto de cada uma escreve-se o endereço Xn.
6. Xn é incrementado [12].
7. Sub1 é encerrada.

[11] Se a mesma palavra w aparece em mais de um item, Sub1 não precisa necessariamente fazer perguntas referentes a cada um desses itens. Ela só precisa fazer perguntas até receber uma resposta "sim".

[12] Obviamente Xn deve ter um valor inicial, um valor para quando o banco de dados ainda está vazio. Mas isso deve ser definido pelo programador.

É com base nessas operações que Sub1 realiza o processo de organização de informação. Aqui já temos o esquema básico do processo; é preciso, no entanto, esclarecer um pouco melhor a função das sub-rotinas R1 e R2.

A sub-rotina R1 é um provador automático. Ela deve transformar as informações do banco de dados em premissas e, a partir dessas premissas, tentar provar um enunciado S (ou S') e sua negação. Aqui, pode haver um problema prático para a execução da tarefa. Na prática, trabalhar com uma grande quantidade de premissas pode ser proibitivo para R1. Alternativamente, o que a rotina pode fazer é gerar os encadeamentos máximos dos itens selecionados no passo 1, utilizar os itinerários correspondentes a esses encadeamentos para localizar informações no banco de dados, transformar essas informações em premissas e usar essas premissas para tentar provar S (ou S') e sua negação. Essa alternativa, por mais complexa que pareça, pode ser a mais viável em muitos casos. Outra coisa que deve ser observada é que, embora R1 gere uma prova a partir de um certo conjunto de premissas, é possível que nem todas as premissas contribuam efetivamente para a prova. No caso específico do teste de consistência, somente as premissas que efetivamente contribuem para a prova de ¬S (ou ¬S'), serão consideradas como "envolvidas na inconsistência".

Para R2 resolver um problema de inconsistência, ela deve avaliar os pesos das informações envolvidas na inconsistência e, além disso, deve verificar se a inconsistência é ocasionada por uma contradição direta entre S e uma das premissas utilizadas por R1. Se não houver uma contradição direta, dentre todas as informações envolvidas na inconsistência, R2 vai apagar ou desconsiderar a informação de menor peso. Se houver empate entre os pesos da informação fornecida por S e de alguma das informações já registradas, R2 vai desconsiderar a informação fornecida por S. Se houver uma contradição direta, dentre as duas informações contraditórias, R2 vai apagar ou desconsiderar a informação de menor peso. Se houver empate entre os pesos, R2 vai desconsiderar a informação fornecida por S. A ideia é que se R2 não tem base para preferir uma informação à outra, por uma questão de economia de processamento, ela deve conservar a que já está registrada e desconsiderar a nova.

Com isso, concluo a descrição de Sub1 e suas rotinas. É possível, porém, que alguns pontos da descrição tenham ficado obscuros. Para esclarecê-los, vou dar alguns exemplos.

Digamos que Sub1 recebe o enunciado “Sócrates foi o mestre de Platão” e que ele passa no TESTE1 e no TESTE2. Digamos também que no passo 1, Sub1 localiza as palavras “Sócrates” e “mestre”, mas não localiza a palavra “Platão”. Nesse caso, que é bem simples, Sub1 passa rapidamente por 3 e vai logo para os passos 4 e 5, registrando a informação do enunciado, atualizando os itens que contêm “Sócrates” e “mestre”, e abrindo um item para a palavra “Platão”. Em seguida, Sub1 passa por 6, chega em 7 e é encerrada aí.

Agora, digamos que depois de processar o enunciado do parágrafo anterior, Sub1 recebe o enunciado “Sócrates era bárbaro” e que a palavra “bárbaro” é classificada como não localizada, embora, vamos supor, exista um item I_b que contém essa palavra. Se a palavra é classificada como não localizada é porque, no enunciado “Sócrates era bárbaro”, a palavra não tem o sentido que tinha nos enunciados relacionados a I_b (pode ser, por exemplo, que nos enunciados relacionados a I_b a palavra “bárbaro” tivesse o sentido de “não grego” e que em “Sócrates era bárbaro” a palavra tenha o sentido de “incrível”). Destarte, Sub1 deve mais uma vez passar rapidamente por 3 e daí deve seguir para os passos 4 e 5, registrando a informação do enunciado, atualizando o item que contêm a palavra “Sócrates” e abrindo um novo item para a palavra “bárbaro”. Em seguida, Sub1 passa por 6, chega em 7 e é encerrada aí.

Observe que mesmo que as informações de que *Sócrates era grego* e de que *gregos eram não bárbaros* já estivessem registradas no banco de dados, não surge nenhum problema de inconsistência com o novo enunciado, pois, uma vez que a palavra “bárbaro” é classificada como não localizada, no TESTE2, R1 vai tentar provar o enunciado “Sócrates era não bárbaro*” (note que o *string* ‘bárbaro*’ é diferente do *string* ‘bárbaro’, e por isso será tomado como uma palavra diferente de “bárbaro”) a partir das premissas “Sócrates era grego” e “gregos eram não bárbaros”, e esse enunciado não pode ser provado a partir dessas premissas.

Digamos então que, na sequência, Sub1 recebe o enunciado “Sócrates é jogador de futebol” e localiza todas as palavras menos a palavra “Sócrates”. Temos aqui a mesma situação do parágrafo anterior, só que agora a palavra não

localizada é um nome próprio. Sub1 fará as mesmas operações que fez no caso anterior, só muda a informação que é registrada, os itens que são atualizados e a palavra que ganha um homônimo. Digamos, porém, que logo em seguida o programa recebe o enunciado “Sócrates é alto”. Dessa vez vamos dizer que todas as palavras são classificadas como localizadas e o item selecionado que contém a palavra “Sócrates” é o item recém-aberto. O que acontece nesse caso também é simples. A informação fornecida pelo enunciado “Sócrates é alto” é registrada em um endereço Xn do banco de dados e o endereço Xn é incluído nos itens selecionados no momento em que Sub1 recebe o enunciado (note que o item que vai ser associado à informação de que *Sócrates é alto* é o mesmo que está associado à informação de que *Sócrates é jogador de futebol* e é diferente do item que está associado à informação de que *Sócrates foi o mestre de Platão*).

Por último, vamos supor que o próximo enunciado que Sub1 recebe é “Sócrates era bonito” e que as palavras componentes do enunciado são localizadas, sendo que a palavra “Sócrates” é localizada em um item relacionado à informação de que *Sócrates era feio* e a palavra “bonito” é localizada em um item relacionado à informação de que *nenhum feio é bonito.* Nesse caso, o enunciado não passa no teste de consistência e R2 é chamada. Tudo agora depende dos pesos associados às informações envolvidas na inconsistência. Se a informação de que *Sócrates era feio* tiver peso menor que as outras, ela será apagada do banco de dados e o endereço que fazia referência a ela será apagado de todos os itens em que ele aparece. E o mesmo acontece com a informação de que *nenhum feio é bonito* se ela tiver peso menor. Se a informação de que *Sócrates era bonito* tiver peso menor ou igual ao peso da informação que tem menor peso entre as outras informações envolvidas na inconsistência, ela será desconsiderada.

Acredito que esses exemplos são suficientes para esclarecer os pontos mais complicados do processo de organização da informação implementado por Sub1. Com a descrição desse processo concluo a descrição de Sub1. Agora vem Sub2.

Sub2 é a segunda parte de ProgX. A parte que efetivamente responde a questão “quem é X?”, para um nome próprio X qualquer. Ora, para responder essa pergunta Sub2 faz uma operação bem simples. Ela entra no índice de endereçamento e procura o nome X. Se X aparece apenas em um certo item I_x, Sub2 acessa os endereços listados em I_x, lê as informações escritas em cada

endereço e dá como resposta a lista de enunciados que expressam aquelas informações. Essa lista de enunciados que está associada a um uso do nome X chamarei de “inventário”.

Se X aparece nos itens I_1,... e I_n, Sub2 acessa os endereços listados em cada item, lê as informações escritas em cada endereço e dá a seguinte resposta: “Existem n X. Sobre o primeiro X é verdade dizer que LISTA1. Sobre o segundo X, é verdade dizer que LISTA2. [...]. Sobre o n-ésimo X, é verdade dizer que LISTAn”. Aqui, o simbolismo LISTAj abrevia um inventário relativo ao item I_j. A resposta de Sub2, portanto, apresentará n inventários diferentes, cada um dos quais corresponde a um uso diferente do nome X.

Finalmente, se X não aparece em nenhum dos itens do índice de endereçamento, Sub2 dá a resposta: “Não sei quem é X”.

Essas são todas as situações que Sub2 encontrará e todas as respostas que ela dará em cada situação. É basicamente assim que a rotina funciona.

É bom lembrar que, embora seja uma rotina independente, Sub2 sempre vai precisar de Sub1, uma vez que é Sub1 que gera o banco de dados e o índice de endereçamento que Sub2 consulta para dar suas respostas. Tanto o banco de dados quanto o índice de endereçamento vão variar com o tempo, e isso pode fazer com que Sub2 responda a mesma pergunta de formas diferentes em momentos diferentes.

Uma vez que Sub1 e Sub2 têm sido descritas, podemos considerar que a descrição de ProgX está completa. Com base nessa descrição, um programa poderia teoricamente ser escrito e implementado em um computador, e esse computador seria capaz de responder a pergunta “quem é X?”. Não posso dizer se um computador teria recursos de memória e velocidade suficientes para realizar as operações descritas acima. Em particular, não sei se as operações de inferência concernentes a R1 poderiam ser implementadas de maneira eficiente.

De todo modo, o que importa aqui é a tese de que nós realizamos procedimentos semelhantes aos de ProgX quando lidamos com nomes próprios. Nós realizamos operações de ponderação, de armazenagem, de classificação e de associação de informações, operações de desambiguação, de inferência, de revisão de crenças etc. Em suma, a descrição de ProgX nos dá um modelo de como a mente humana processa informações fornecidas por enunciados que mencionam um nome próprio X. É verdade que ProgX lida apenas com enunciados

declarativos e perguntas, mas algumas das operações que ele implementa são fundamentais para a implementação de qualquer processo de decodificação de linguagem e nomes próprios. Inclusive para a implementação do processo de decodificação levado a efeito pela mente humana.

Poder-se-ia argumentar que nós não podemos ser comparados a um programa de computador. Que nós fazemos as coisas de modo diferente ou, pelo menos, de modo parcialmente diferente. Bem, eu não nego que o processamento linguístico humano tem suas especificidades. Meu objetivo é apenas mostrar que as operações mais gerais são semelhantes. Ninguém discute, por exemplo, que, quando precisamos decidir sobre a credibilidade de uma informação, tipicamente julgamos que as informações mais críveis são as que provêm das fontes mais idôneas e respeitáveis. Por exemplo, em geral, uma informação sobre Sócrates dada por um renomado estudioso da história da filosofia antiga terá mais peso que uma dada por um iniciante no estudo da filosofia. É verdade que podemos nos enganar sobre a credibilidade de nossas fontes e que há fatores emocionais e inconscientes que podem influenciar nossos julgamentos, mas esse não é o ponto. O ponto é que fazemos operações muito semelhantes as que R2 faz. Da mesma forma que R2, nós também precisamos considerar os pesos que damos às informações para resolver situações de inconsistência, e, da mesma forma como acontece no nosso modelo, esse peso tem relação com a fonte da informação. Isso ajuda a mostrar que, no caso geral, os processos mentais que implementamos quando usamos nomes próprios realizam tarefas semelhantes àquelas de ProgX.

Alguém poderia alegar que não fazemos o que Sub1 faz no passo 1 do processo de organização de informação. Mas a verdade é que nós também buscamos informação adicional para reconhecer a forma como as palavras são usadas. Acontece que nós podemos obter essa informação adicional de fontes bem variadas, enquanto que o programa só pode obter informação dos seus operadores. O contexto em que uma palavra é usada fornece grande parte da informação adicional que usamos e, quando, essa informação ainda não é suficiente para esclarecer o modo como as palavras estão sendo usadas, nós fazemos exatamente o que Sub1 faz, nós fazemos perguntas ao nosso interlocutor visando esclarecer esses usos.

De um modo geral, ProgX nos oferece muitas analogias esclarecedoras. Mas duas coisas que o modelo esclarece têm que ser colocadas em

destaque. Em primeiro lugar, o modelo confirma que nomes próprios, como qualquer expressão da linguagem, têm fundamentalmente uma função operacional. Quando Sub1 recebe um enunciado e nesse enunciado ocorre um nome próprio n, n aciona certas operações na rotina. Especificamente, o nome aciona operações de busca e de desambiguação. E essas operações só podem ser executadas porque o nome é associado a certas informações do banco de dados. De fato, essa é a segunda coisa que o modelo esclarece e que merece um destaque especial, a saber, um nome sempre está relacionado a uma lista de informações. Isso é demonstrado tanto por Sub1, que escreve um nome próprio n no índice de endereçamento e o relaciona com informações no banco de dados, como por Sub2, que acessa essas informações e gera um inventário para cada uso de n.

Deve-se observar que essas duas coisas que ProgX esclarece são bem diferentes. Uma coisa é a informação que n fornece ao programa para que as operações apropriadas comecem a ser executadas, outra coisa é a informação que o programa, ao final do processo, associa a n. Quando n é introduzido em ProgX, a informação que ele fornece é inicialmente uma informação-para, ela funciona como um gatilho, é uma informação que dispara processos, é uma informação que poderíamos chamar também de “informação operacional”. Mas isso não é só com os nomes próprios. Todas as palavras que são introduzidas em ProgX funcionam assim. Depois que esses processos são disparados e completados é que as palavras são associadas às informações-sobre fornecidas pelos enunciados introduzidos em Sub1. Como essas informações advêm de enunciados declarativos, poderíamos chamá-las também de “informações proposicionais”. O modelo mostra que isso acontece com todas as palavras, inclusive com os nomes próprios.

Mas, mais importante do que observar o que ProgX faz, é perceber que nós fazemos coisas muito semelhantes. Parece razoável admitir que se nomes próprios funcionam como códigos quando são processados por ProgX, eles também funcionam como códigos quando são usados por nós. E, outrossim, parece razoável concluir que se nomes próprios são associados a informações proposicionais quando são processados por ProgX, eles também são associados a informações proposicionais quando são usados por nós.

Por fim, gostaria de chamar a atenção para o fato de que todas as operações que ProgX executa ao receber um nome próprio são executadas sem se dar a menor atenção ao fato do nome ser usado referencialmente ou não. ProgX

processaria o nome “Oswaldo” da mesma forma que ele processaria o nome “Apolo”. Isso mostra que um nome próprio não precisa ter uma função referencial para ter uma função operacional.

Em contrapartida, no meu modo de ver, não é possível que um nome desempenhe uma função referencial se antes não desempenhar uma função de código. No próximo capítulo, uma das coisas que vou fazer é tentar esclarecer essa dependência da função referencial em relação à função operacional. Para isso, mais uma vez vou me valer de algumas coisas que ProgX esclarece. Especificamente, a ideia de que nomes próprios sempre são associados a informações-sobre me será bastante útil.