Pós-Graduação em Ciência da Computação

Arthur Freitas Ramos

TIPO IDENTIDADE COMO O TIPO DE CAMINHOS COMPUTACIONAIS

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2015

Arthur Freitas Ramos

TIPO IDENTIDADE COMO O TIPO DE CAMINHOS COMPUTACIONAIS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Ruy J. G. B. de Queiroz*

RECIFE
2015

Catalogação na fonte
Bibliotecária Joana D'Arc Leão Salvador CRB4-532

R175t Ramos, Arthur Freitas.
Tipo identidade como o tipo de caminhos computacionais / Arthur Freitas Ramos. – Recife: O Autor, 2015.
92 f.: il., quadro.

Orientador: Ruy J. G. B. de Queiroz.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIN, Ciência da Computação, 2015.
Inclui referências.

1. Teoria da computação. 2. Teoria dos tipos. I. Queiroz, Ruy J. G. B. de (Orientador). II. Titulo.

005.131 CDD (22. ed.) UFPE-MEI 2015-100

Dissertação de Mestrado apresentada por **Arthur Freitas Ramos** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Tipo Identidade Como o Tipo de Caminhos Computacionais**", orientada pelo **Prof. Ruy José Guerra Barretto de Queiroz** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Rafael Dueire Lins
Centro de Informática/UFPE

Prof. Eduward Hermann Haeusler
Departamento de Informática/PUC-Rio

Prof. Ruy José Guerra Barretto de Queiroz
Centro de Informática/UFPE

Vista e permitida a impressão.
Recife, 29 de julho de 2015.

Profa. Edna Natividade da Silva Barros

Coordenadora da Pós-Graduação em Ciência da Computação do

Centro de Informática da Universidade Federal de Pernambuco.

Eu dedico essa dissertação aos meus pais. Por toda a ajuda,
o apoio, o amor. Por tudo.

Agradecimentos

Primeiro, agradeço a Deus por ter me dado todas as oportunidades e a resiliência necessária para chegar até aqui.

Agradeço infinitamente aos meus pais por terem sido heróis a vida toda, trabalhando arduamente para que eu não tivesse que me preocupar praticamente com nada além dos meus estudos. Obrigado pelo suporte, pelo carinho, por serem os melhores pais que alguém poderia pedir. Me sinto abençoado por ser filho de vocês.

Agradeço a minha namorada Cássia por, nesses 18 meses de mestrado, sempre me apoiar completamente, me ajudar a tomar decisões dificéis e entender as limitações que a vida de mestrando tem, além de ser a principal pessoa a me dar forças sempre que eu encontrava algum obstáculo. Esse trabalho não seria possível sem você.

Gostaria de agradecer aoes meus grandes amigos: Davi Duarte (obrigado pelo eterno trabalho de comunicação), Tullio Lucena (tantas coisas para agradecer da graduação, que nem vou começar a listar...), Huguinho Neiva (que também está se aventurando nessa vida de mestrando), Dalton Santama (vê se aparece!) e David Hulak (virando um capitalista na Microsoft). Amigos que, sem dúvida alguma, ficarão para toda uma vida.

Gostaria de agradecer a minha irmã (e nosso amigo Vinicius!) por ter as melhores soluções para resolver o estresse, mostrando que uma ou duas cervejas em uma quinta-feira estressante faz bem a longo prazo.

Por fim, também não tenho palavras para agradecer a ajuda do meu orientador Ruy de Queiroz. Primeiro que a escolha de um tema tão fantástico como o desse trabalho seria impossível sem a presença dele. Segundo, por ter dado toda a ajuda necessária e ter me apontado os possíveis caminhos a se seguir, me dando a liberdade para escolhê-los. Obrigado pela flexibilidade, por ter me poupado de diversas burocracias que só fazem atrasar o desenvolvimento do conhecimento. Espero que possamos continuar essa ótima experiência nos possíveis próximos 3 anos de Doutorado.

There is nothing outside of yourself
that can ever enable you to get
better, stronger, richer, quicker or smarter.
Everything is within. Everything exists.
Seek nothing outside of yourself.

—MYAMOTO MUSASHI

Resumo

O presente trabalho tem como objetivo o estudo de uma entidade computacional conhecida como caminhos computacionais. Proposta originalmente por QUEIROZ; OLIVEIRA (2011) como 'sequências de reescritas', a ideia é que os caminhos computacionais funcionam como os termos do tipo identidade da Teoria dos Tipos de Martin-Löf'. Esse trabalho expande essa ideia, mostrando uma formalização completa do tipo identidade a partir dos caminhos computacionais. É mostrado que os caminhos computacionais tornam o tipo identidade uma entidade muito mais intuitiva e simples, quando comparado com a abordagem tradicional.

Um outro foco desse trabalho é o estudo das propriedades matemáticas dos caminhos computacionais. Em particular, o interesse é em explorar a relação entre os caminhos e a teoria das categorias. Especificamente, é provado que os caminhos computacionais são capazes de induzir uma estrutura algébrica conhecida como grupóide. Esse resultado está de acordo com os resultados obtidos por HOFMANN; STREICHER (1994), que mostram que a abordagem tradicional do tipo identidade induzem um modelo de grupóide.

Palavras-chave:

Tipo identidade. Caminhos computacionais. Teoria dos tipos. Teoria da igualdade. Construções a partir de caminhos. Sistema de reescrita de termos. Modelo de grupóide. Teoria das categorias.

Abstract

The current work aims to study a computational entity known as computational paths. Originally proposed by QUEIROZ; OLIVEIRA (2011) as 'sequence of rewrites', the idea is that computational paths are terms of Martin-Löf's identity type. This work expands this idea, showing a complete formalization of the identity type using computational paths. It is shown that computational paths makes the identity type a much simpler and more intuitive entity, when compared to the tradional approach.

Another focus of this work is the study of the mathematical properties of computational paths. In particular, the main aim is the relation between computational paths and category theory. Specifically, this work shows that computational paths are capable of inducing an algebraic structure known as groupoid. This results is on a par with the one obtained by HOFMANN; STREICHER (1994), which shows that the traditional approach of the identity type also induces a grupoid model.

Keywords: Identity type. Computational paths. Type theory. Equality theory. Path-based constructions. Term rewriting system. Groupoid model. Category theory.

List of tables

Lista de Quadros

Lista de Acrônimos

Sumário

1

Introdução

Este capítulo tem dois objetivos principais. O primeiro é introduzir o tipo identidade como ponte entre os fundamentos da computação e os da matemática. Para isso, será necessário fazer uma breve contextualização sobre os fundamentos atuais de ambas as áreas. Além disso, será necessário mostrar os problemas causados pela teoria axiomática dos conjuntos, a qual é a teoria que é, atualmente, a mais amplamente utilizada como fundamento da matemática. Com esses breves tópicos cobertos, o tipo identidade será devidamente introduzido e sua importância ficará evidente. O segundo objetivo a ser atingido é indicar quais resultados este trabalho pretende atingir. Os resultados irão envolver diretamente o tipo identidade, o qual será o principal objeto de estudo deste trabalho.

1.1 Fundamentos da Matemática

O século XIX foi responsável pela forma abstrata de pensamento matemático, conhecido também como forma moderna do pensamento matemático (AVIGAD, 2007). Antes desse século, a produção matemática era baseada em computações e construções. Porém, com o aumento progressivo da abstração, a matemática se distanciou dessa pratica mais algorítmica (AVIGAD, 2007). Entre os avanços que mais contribuíram para o aumento da abstração, estão descobertas extremamente importantes, como a existência de geometrias não-euclideanas. Exemplos de tais geometrias são a geometria hipérbolica, criada por Nikolai Lobachevsky em 1832 e a geometria elíptica, criada por Bernhard Riemann em 1851.

A crescente abstração da matemática, mencionada anteriormente, teve como uma das consequências o crescente interesse na chamada filosofia da matemática. A filosofia da matemática é uma área da filosofia que procura interpretações filosóficas para as origens de um objeto matemático abstrato. Entre as várias interpretações, as mais aceitas são duas: o platonismo e o construtivismo. O platonismo, o qual é a teoria mais aceita pela comunidade matemática atual, é baseada na ideia de que existe um mundo abstrato e imutável que contém todos os objetos matemáticos. Dessa forma, para essa teoria, o matemático nunca constrói um elemento abstrato novo, mas apenas descobre um objeto que ainda não tinha sido descoberto anteriormente,

mas que já existia no universo platônico dos objetos da matemática. Já para o construtivismo, um objeto matemático passa a existir a partir do momento em que o matemático o constrói mentalmente. Como pode-se perceber, dada a visão algorítmica e construtiva de um programa de computador, o qual é construído a partir de processos algorítmicos, os objetos computacionais se encaixam na visão construtivista da matemática.

A escolha entre construtivismo e platonismo não tem apenas implicações filosóficas, mas também implicações práticas que influenciam a forma de fazer demonstrações na matemática. Um exemplo simples, mas que demonstra essa afirmação, são as diferentes provas para a seguinte proposição:

Proposição 1.1. *Existem números irracionais a e b tal que a^b é um número racional.*

Demonstração: Considere o número $\sqrt{2}$, que é claramente irracional. Logo, $\sqrt{2}^{\sqrt{2}}$ é irracional ou racional. Se o número é racional, a prova esta concluída. Se não for, $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^2 = 2$, também concluindo a prova, já que foi considerado que $\sqrt{2}^{\sqrt{2}}$ é irracional. □

Essa prova mostra que um dos dois casos apresentados resolve o problema, mas não permite distinguir qual deles é o que resolve. Por isso, essa demonstração não é construtiva (DUMMETT, 1977). Segundo o ponto de vista platônico, porém, essa demonstração é completamente aceitável. Considere, agora, a seguinte demonstração alternativa:

Demonstração: Considere os números $\sqrt{2}$ e $\log_{\sqrt{2}} 3$. Como ambos são irracionais e $\sqrt{2}^{\log_{\sqrt{2}} 3} = 3$, que é um número racional, a demonstração esta concluída. □

Essa demonstração é diferente da anterior, já que a construção de um exemplo é claramente explicitada. Portanto, caracteriza-se como uma prova *construtiva*. Quando se fala em computação, os únicos tipos de provas (de existência) aceitáveis são as que são como a do segundo caso, devido à natureza construtiva e algorítmica de um programa de computador.

Junto com essa discussão envolvendo a origem dos objetos matemáticos, outro assunto de grande interesse para a comunidade matemática do final do século XIX e início do século XX foi a procura de uma base sólida para os fundamentos da matemática. O estudo dos fundamentos da matemática tem como objetivo caracterizar a natureza básica dos objetos matemáticos. É a tentativa de achar uma base para matemática, a qual justifica a existência e o conhecimento de entidades matemáticas (HORSTEN, 2015). Entre as tentativas de achar um fundamento para a matemática, destaca-se o logicismo de Frege, criado em 1884 com a publicação do livro *Die Grundlagen der Arithmetik*.

O logicismo foi uma escola que tentou estabelecer os fundamentos da matemática através da lógica. A ideia era que toda a matemática podia ser derivada a partir de princípios lógicos. Frege, por exemplo, conseguiu derivar, a partir da lógica, a forma mais clássica da aritmética, que é a aritmética utilizando os axiomas de Peano (HORSTEN, 2015). Para isso, Frege utilizava extensivamente uma propriedade da lógica que dizia que para qualquer propriedade

matemática, existe uma classe de identidades que possuem tal propriedade (HORSTEN, 2015). Essa propriedade, ao mesmo tempo que tinha um imenso poder de uso, gerou um paradoxo que levou ao rápido fim do logicismo.

Em 1901, o filósofo e lógico Bertrand Russell descobre um paradoxo que, como consequência, causou o rápido abandono do logicismo pela comunidade matemática. Utilizando os princípios do logicismo, é possivel criar a seguinte classe de entidades: $R = \{x | x \notin x\}$. Russell notou que isso gerava um paradoxo (IRVINE, 2014), pensando da seguinte forma: Se R não é membro dele mesmo, então, pela definição de R, ele teria que ter a si próprio como um membro. Mas R não pode ser membro dele mesmo, pois entra em contradição com a definição de R. Dessa forma, R não pode existir.

A ideia que a existência de um conjunto de todos os conjuntos que não têm a si próprios como membro leva a um paradoxo, foi um grande golpe nao só ao logicismo, mas também ao modo de se trabalhar com conjuntos. Ficou clara a necessidade de se criar um conjunto de regras que definiam como os conjuntos podiam ser criados e utilizados. Esse foi o objetivo do matemático Ernest Zermelo ao propor, em 1908, um sistema axiomático que se tornou a base para a teoria axiomática do conjuntos. Esse sistema, junto com vários melhoramentos ocorridos no período entre 1920-1940, inclusive com uma segunda axiomatização proposta por Zermelo, deu origem à teoria axiomática dos conjuntos atual, conhecida como Zermelo Frankel com axioma da escolha (ZFC) (HALLETT, 2013).

O sucesso do ZFC foi tão grande que até hoje é utilizado por grande parte da comunidade matemática como o fundamento da matemática. Ou seja, as entidades matemáticas complexas podem ser formalizadas a partir dos axiomas básicos contidos no ZFC. Entretanto, com o constante aumento da complexidade da matemática, o ZFC tem se revelado cada vez mais problemático. O que reforça esse fato é que o ZFC é não-construtivo, muito embora algumas variantes construtivas tenham sido formuladas e analisadas. Portanto, há uma certa incompatibilidade profunda na tentativa de modelar o ZFC usando computadores. Com a matemática ficando cada vez mais complexa, é fato que que o uso de computadores para auxiliar na checagem das provas está se tornando cada vez mais essencial. Na próxima seção, serão destacados os problemas do ZFC e a tentativa de adotar uma nova teoria que sirva como fundamento para matemática, que possa ser modelada através de computadores e, ocorre que uma das alternativas recentemente propostas tem o tipo identidade como principal entidade.

1.2 Os Problemas do ZFC

Por mais de cem anos, o ZFC foi utilizado com sucesso como fundamento da matemática. O objetivo desta seção é mostrar que o ZFC possui duas características básicas que impedem que ela seja modelado por computadores: a não-construtividade e o extensionalismo. Para explicar essas duas caracteristicas, serão utilizados alguns conceitos de teoria dos conjuntos, mas nada muito complexo.

Um dos conceitos mais importantes do ZFC e que sera utilizado para mostrar ambas as caracteristicas é o conceito de relação:

Definição 1.1 (Relação (HRBACEK; JECH, 1999)). *Um conjunto R é uma relação binária se todos os elementos de R são pares ordenados.*

Pegando todos os primeiros elementos de todos os pares ordenados, obtém-se um conjunto chamado $domR$ (domínio da relação R). Pegando os segundos elementos, obtém-se o conjunto imR (imagem da relação R). Um par ordenado $(a,b) \in R$ também pode ser representado da forma aRb.

Com o conceito de relação em mãos, é possivel definir o conceito de função:

Definição 1.2 (Função (HRBACEK; JECH, 1999)). *Uma relação binária F é uma função se aFb_1 e aFb_2 implica $b_1 = b_2$ para qualquer a, b_1 e b_2.*

Olhando atentamente, essa definição coincide com a definição mais comum de função, em que cada elemento do domínio tem apenas um elemento correspondente na imagem da função. Os conceitos de domínio e imagem coincidem com os conceitos anteriormente definidos de domínio e imagem de relações. Não poderia ser diferente, já que uma função é apenas uma relação que obedece propriedades adicionais.

É exatamente essa definição que deixa claro a extensionalidade do ZFC. No ZFC, uma função é basicamente um conjunto de pares de entrada e saída que obedece a propriedade citada anteriormente. Portanto, dado uma entrada qualquer, a única coisa importante é a saída. O processo de como essa saída foi obtida é irrelevante. Dadas duas funções, elas podem ter funcionamentos internos completamente diferentes, mas caso elas tenham as mesmas entradas e saídas, elas são consideradas iguais de acordo com o ZFC. Por não considerar o processo em que a saída é gerada a partir de uma entrada, é dito que essa teoria é extensional.

Compare agora essa definição com a definição computacional para função. Dado um problema, existem diversas funções (algoritmos) que o resolvem. Porém, geralmente esses algoritmos usam técnicas diferentes e, muitas vezes, possuem complexidade de tempo e espaço distintas. Portanto, do ponto de vista computacional, esses algoritmos são distintos, apesar de resolver o mesmo problema. Por esse motivo, o modo através do qual uma saída é obtida a partir de uma entrada é essencial. Portanto, pode-se dizer que a computação é uma teoria de funções cuja igualdade é intensional. Será visto mais adiante que o tipo identidade traz no seu bojo as ferramentas apropriadas para simular essa intensionalidade da relação de igualdade entre funções.

Sendo a intensionalidade uma propriedade essencial para qualquer teoria da computação, já fica claro um dos motivos porque o ZFC não se apresenta como o melhor candidato para servir de base aos fundamentos da computação. O outro motivo, o qual será explicado a seguir, é o fato que o ZFC tem uma natureza não-construtiva. Isso fica claro quando o axioma da escolha passa a ser utilizado. Cabe acrescentar, no entanto, que há versões construtivas da teoria de conjuntos,

como, por exemplo, a Teoria Construtiva de Conjuntos de Myhill, que busca reformular alguns dos axiomas de ZFC com vistas a dar origem a teoremas somente sob os critérios da matemática construtiva.

O axioma da escolha é um dos mais polêmicos axiomas encontrados na matemática. O motivo disso é que a utilização desse axioma gera alguns resultados não intuitivos. Um desses resultados será utilizado para deixar claro a não-construtividade do ZFC. Antes, porém, é preciso definir alguns conceitos básicos:

Definição 1.3 (Antissimetria (HRBACEK; JECH, 1999))**.** *Uma relação binária R em um conjunto A é antisimmétrica se para todo $a, b \in A$, aRb e bRa implica em $a = b$.*

Definição 1.4 (Ordenação de A (HRBACEK; JECH, 1999))**.** *Uma relação binária R em A que é reflexiva, antissimétrica e transitiva é chamada de ordenação de A. O par (A, R) é chamado de conjunto ordenado.*

Esses conceitos formalizam a ideia de ordenação, utilizando o ZFC. Fica claro que as ordenações usuais de conjuntos como $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ e $\mathbb{R}$ seguem essa definição. Indo mais adiante, tem-se a seguinte definição:

Definição 1.5 (Ordenação linear de A (HRBACEK; JECH, 1999))**.** *Uma ordenação $\leq$ de A é chamada linear ou total se é possível comparar quaisquer dois elementos de A. O par $(A, \leq)$ é chamado de conjunto linearmente ordenado.*

Não tem nada que obrigue, na definição usual de ordenação, que quaisquer dois elementos possam ser comparados. Por isso, torna-se necessária essa definição de ordenação linear. Como exemplo, pode-se usar os conjuntos citados anteriormente, junto com as respectivas ordenações usuais. Pode-se, por exemplo, comparar quaisquer dois elementos dos conjuntos dos naturais ou dos reais. Por fim, a seguir tem-se a última definição envolvendo ordem:

Definição 1.6 (Conjunto bem-ordenado (HRBACEK; JECH, 1999))**.** *Um conjunto W é bem ordenado pela relação $<$ se:*

- *$(W, <)$ é um conjunto linearmente ordenado*
- *Todo subconjunto não-vazio de W tem um menor elemento.*

O exemplo clássico de um conjunto bem-ordenado é $(\mathbb{N}, \leq)$, com $\leq$ sendo a ordenação usual. Como os naturais não possuem números negativos, fica fácil ver que qualquer subconjunto terá um menor elemento. Por outro lado, a ordenação usual de $\mathbb{Z}$ não é uma boa-ordenação, porque basta pegar qualquer subconjunto que contém os infinitos números negativos. Esse subconjunto não terá menor elemento. Seria possível, então, achar uma boa-ordenação para $\mathbb{Z}$? A resposta é positiva. Segue a prova:

Demonstração: Considere uma relação $<'$ em que o primeiro elemento é 0 e todos os números positivos são menores que os números negativos. Além disso, ao comparar dois números negativos ou dois números positivos, $<'$ usa a ordenação usual. Dessa forma, $<'$ ordena $\mathbb{Z}$ da seguinte forma: $0, 1, 2, 3... - 1, -2....$ Dessa forma, qualquer subconjunto terá um menor elemento. □

É nesse contexto de conjuntos bem-ordenados que será mostrado a não construtividade do axioma da escolha. O axioma da escolha aparece de diversas formas equivalentes na matemática. A que será usada aqui é uma dessas formas, que é possível mostrar que é equivalente à forma original:

Proposição 1.2. *Todo conjunto pode ser bem-ordenado.*

Como dito anteriormente, pode ser provado que o axioma da escolha é equivalente à proposição acima (HRBACEK; JECH, 1999). Por ser uma prova relativamente longa e que não adiciona nada relevante, ela será omitida.

A parte surpreendente da proposição anterior é que até conjuntos como os reais podem ser bem-ordenados. O grande problema, porém, é que ninguém conseguiu construir uma ordenação que mostra explicitamente que os reais podem ser bem ordenados, apesar de o axioma da escolha garantir isso. Portanto, o axioma da escolha repousa sobre pressupostos platonistas, já que diz que um objeto existe, sem explicitar uma construção. Isso mostra que, fundamentalmente, o ZFC é uma teoria não-construtiva.

Com isso, mostramos que o ZFC não se apresenta como fundamentos apropriados a uma teoria de algoritmos. A busca por uma teoria que seja intensional, construtiva e que ainda sirva como base para os fundamentos da matemática encontra regozijo na descoberta do tipo identidade como ponte entre os fundamentos da computação e da matemática, como será visto na proxima seção.

1.3 O Tipo Identidade e a Ligação Entre Computação e Matemática

Nessa seção, argumentamos que a ligação entre uma teoria da computação com a matemática pode ser estabelecida através do tipo identidade. O tipo identidade é, possivelmente, a principal entidade de uma teoria da computação chamada teoria dos tipos. A teoria dos tipos é uma teoria construtiva criada pelo matemático Per Martin-Löf em 1971 (MARTIN-LÖF, 1975, 1982; MARTIN-Löf, 1984). Nessa teoria, o conceito fundamental é o tipo. Todo objeto matemático é um tipo e um tipo é definido a partir da descrição de o que deve-se fazer para construí-lo (BRIDGES; PALMGREN, 2013). A teoria será desenvolvida com detalhes no proxímo capítulo. Nessa seção, o foco será mostrar a importância do tipo identidade e dos caminhos computacionais.

Dado um determinado tipo A, denota-se $a : A$ para indicar que a é um objeto do tipo A. O tipo identidade é o que permite que a teoria dos tipos torne-se intensional. O tipo identidade

encapsula a seguinte ideia: dado objetos $a : A$ e $b : B$ e um conjunto de passos p que estabelecem que $a = b$, então é dito que p é um elemento do tipo identidade $Id_A(a,b)$. Ou seja, os objetos do tipo $Id_A(a,b)$ são provas que estabelecem que $a = b$. Dessa forma, é possível dar sentido intensional para o tipo identidade, já que ele encapsula não só a informação que $a = b$, mas também o motivo que estabelece essa igualdade.

A teoria dos tipos ganhou grande visibilidade na comunidade matemática após a descoberta de uma relação direta entre o tipo identidade e a homotopia (VOEVODSKY, 2014). Nessa interpretação, um tipo A é visto como um espaço topológico, os objetos $a, b : A$ são vistos como pontos do espaço e um elemento $p : Id_A(a,b)$ é visto como um caminho homotópico entre os pontos a e b (UNIVALENT FOUNDATIONS PROGRAM, 2013). Essa interpretação, a qual é relativamente simples, tem resultados surpreendentes. Um dos mais importantes é que essa ligação matemática entre homotopia e teoria dos tipos através do tipo identidade cria a possibilidade de considerar a teoria dos tipos como fundamento da matemática, substituindo o ZFC. Por ser intensional e construtiva, é possível utilizá-la como base para a formalização dos processos que se realizam em computadores.

A vantagem de utilizar uma teoria que pode ser modelada através de computadores como fundamento da matemática é veementemente defendida pelo matemático Vladimir Voevodsky (VOEVODSKY, 2014). Voevodsky, o qual é um respeitado matemático agraciado com a Medalha Fields, é o principal responsável pela descoberta entre a ligação da teoria dos tipos com a teoria da homotopia. Voevodsky argumenta que o aumento da abstração e complexidade na matemática torna inviável provar novos teoremas e resultados sem a ajuda de um checador de provas automático. Ele baseia essa necessidade na própria experiência: um artigo publicado por Voevodsky em 1989 contendo resultados envolvendo uma entidade chamada ∞-grupóide serviu como base para diversos resultados posteriores. O problema é que Voevodsky só veio descobrir que o artigo continha um erro crítico em 2013 (VOEVODSKY, 2014). Caso a matemática fosse desenvolvida com ajuda de um checador de provas automático, erros críticos como esse deixariam de existir. Como o ZFC é inviável de ser modelado computacionalmente, Voevodsky destaca a importância de utilizar a teoria dos tipos como novo fundamento da matemática. Tudo isso possível pela simples conexão já mencionada entre o tipo identidade e a homotopia.

O tipo identidade, porém, tem uma grande desvantagem. Apesar de ser perfeitamente viável de ser modelado computacionalmente, o tipo identidade, na forma que está definido atualmente, é de difícil manuseio como elemento de construção de provas, sobretudo devido às suas regras (de eliminação) extremamente complexas e não intuitivas. Uma das propriedades principais do tipo identidade é conhecida como indução no caminho (UNIVALENT FOUNDATIONS PROGRAM, 2013). A indução no caminho é extensivamente usada em diversas provas matemáticas, sejam elas simples ou complexas. O grande problema é que, até nas provas simples, como provas da propriedade de simetria e transitividade, a indução no caminho torna-se complexa e difícil de entender. Os detalhes da indução no caminho e as dificuldades inerentes a esta serão amplamente discutidos no próximo capítulo.

Com o objetivo de simplificar a definição e as propriedades do tipo identidade, uma nova abordagem foi proposta por QUEIROZ; OLIVEIRA (2011). Nessa abordagem, o tipo identidade seria visto de uma forma mais computacional. Um objeto $p : Id_A(a,b)$ passa a ser visto como um *caminho computacional* entre a e b. O caminho computacional seria um termo composto através da aplicação de um conjunto de regras de reescritias com origem no λ-cálculo. Essa abordagem, a qual será bem detalhada no próximo capítulo, será o foco de estudo desse trabalho. O motivo para isso é o fato que, substituindo a abordagem atual por essa baseada em caminhos computacionais, a teoria dos tipos se tornaria extremamente mais simples e intuitiva.

O grande problema enfrentado pelos caminhos computacionais, porém, é a falta de embasamento matemático. Apesar de complexo e não intuitivo, vários resultados matemáticos importantes já foram obtidos a partir da abordagem atual do tipo identidade. Um desses resultados é que o tipo identidade induz uma estrutura conhecida como ω-grupóide (BERG; GARNER, 2011; LUMSDAINE, 2009). Essa estrutura tem propriedades baseadas em uma forma avançada de uma teoria conhecida como teoria das categorias. Além disso, essas propriedades tem grande importância matemática. Portanto, atingir resultados matemáticos semelhantes utilizando o tipo identidade como caminho computacional é o grande desafio.

1.4 Objetivos

Dado o contexto descrito nas seções anteriores junto com a importância da teoria dos tipos tanto para a computação quanto para a matemática, esse trabalho tem dois objetivos principais. O primeiro é estabelecer os caminhos computacionais como entidades básicas do tipo identidade, mostrando as vantagens claras em relação à abordagem atual. O segundo, um pouco mais complexo que o primeiro, é obter resultados matemáticos envolvendo o tipo identidade com a abordagem de caminho computacional, mostrando que resultados matemáticos relevantes podem ser obtidos através dessa nova abordagem.

O primeiro objetivo será obtido através de uma comparação direta entre as duas abordagens. Ao mostrar ambas as abordagens, ficará claro a vantagem, em termos de simplicidade, do tipo identidade como tipo de caminhos computacionais. Além disso, para deixar as vantanges ainda mais claras, serão feitas comparações de provas de termos básicos da teoria dos tipos, como o termo transitivo e o simétrico, utilizando ambas as abordagens.

O segundo objetivo será obtido através de uma análise dos caminhos computacionais através de uma teoria da matemática conhecida como teoria das categorias. Para isso, serão expostos os conceitos necessários dessa teoria para que a interpretação matemática dos caminhos computacionais possa ser bem entendida. Eventualmente, interpretações mais complexas, envolvendo uma área mais avançada da teoria das categorias, serão expostas e explicadas.

Dessa forma, enquanto o primeiro objetivo terá mais base computacional, o segundo terá uma forte base de uma teoria matemática bem estabelecida, como é o caso da teoria das categorias.

1.5 Estrutura da dissertação

Esse primeiro capítulo introdutório foi dedicado a uma introdução da importância do tipo identidade, além de mostrar a abordagem em que o trabalho será focado. Os objetivos que este trabalho pretende atingir também foram expostos.

O segundo capítulo terá como foco a teoria dos tipos e o tipo identidade. Nesse capítulo os conceitos básicos da teoria dos tipos serão expostos, além das diversas abordagens do tipo identidade: extensional, intensional tradicional e intensional como caminho computacional. Detalhes sobre os caminhos computacionais serão explicados e comparações serão feitas com a abordagem tradicional.

O terceiro capítulo terá como foco explicar conceitos de teoria das categorias, incluindo os conceitos básicos e os um pouco mais avançados. Será o capítulo cuja base irá sustentar os resultados obtidos nos capítulos 4 e 5.

O quarto capítulo terá como foco obter resultados matemáticos envolvendo caminhos computacionais e teoria das categorias. Nesse capítulo, a interpretação categórica dos caminhos computacionais será explicitada, usando os conceitos definidos no capítulo 3 juntamente com conceitos intrísicos aos caminhos computacionais. Também será discutida a diferença entre categorias fracas e estritas.

O quinto capítulo será uma expansão dos resultados obtidos no quarto capítulo. Será utilizada uma forma avançada da teoria das categorias, conhecida como teoria das categorias de alta ordem. Será feita uma abordagem utilizando uma estrutura conhecida como bicategoria. Os resultados obtidos serão analisados para conjecturar a construção de uma categoria de infinitas dimensões.

Por fim, o último capítulo será dedicado para as considerações finais. Nesse capítulo, será feita uma rápida revisão dos resultados importantes obtidos. Além disso, as possibilidades de resultados e trabalhos futuros serão discutidas.

2

Teoria dos Tipos e Caminhos Computacionais

Entre as teorias que servem como fundamento da computação, sem dúvida a mais famosa e que mais se assemelha a um computador de propósito geral são as maquinas de Turing. Descrita em 1937 por Alan Turing, as máquinas de Turing obtiveram grande sucesso ao descrever matematicamente o conceito de computador, além de investigar a complexidade de algoritmos e quais problemas podem ser resolvidos computacionalmente (BARKER-PLUMMER, 2013). Entretanto, antes mesmo do trabalho surpreendente de Turing, uma teoria que tem o mesmo poder computacional que a máquina de Turing já havia sido proposta por Alonso Church. A teoria é conhecida como λ-cálculo e é de extrema importância para a computação, lógica e matemática.

Pode-se dizer que o λ-cálculo nasceu com o objetivo de simplificar a notação do conceito de função (ALAMA, 2015). Por isso, a princípio o λ-cálculo parece ser um sistema bastante simples, com apenas duas operações conhecidas como abstração e aplicação. Para explicar a ideia intuitiva dessas operações, pense na função $f(x) = x+2$. Como um valor, $f(2)$ por exemplo, é obtido? Basta substituir x por 2 no termo $x+2$, obtendo-se $2+2=4$. Essa é a ideia que a abstração pretende capturar. Dado um termo $x+2$, abstração λx indica que o termo espera uma entrada que indicará o valor de x. O termo final, então, torna-se o seguinte: $\lambda x.(x+2)$. Já a aplicação é o processo inverso. Dado o termo $\lambda x.(x+2)$, uma aplicação a esse termo serve para substituir o valor de entrada x por um valor específico y. Ao aplicar y, teremos: : $(\lambda x.(x+2))y$. Dizemos que esse termo β-reduz (lê-se beta-reduz) para $[y/x](x+2) = y+2$. $[y/x]$ serve para indicar que todas as ocorrências de x em um termo serão substituídas por y. Essa notação $[y/x]$ também será extensivamente utilizada pela teoria dos tipos.

Por mais surpreendente que possa parecer, é possível provar que essa teoria, a qual tem apenas dois tipos simples de operação, é capaz de representar a própria ideia de número e todas as funções computáveis (HINDLEY; SELDIN, 2008). Por isso, junto com a máquina de Turing, o λ-cálculo é uma das principais teorias que servem como fundamento da computação. Contudo,

Grande parte do capítulo (seção 2.3 em diante) foi baseado em artigo escrito pelo autor dessa dissertação, juntamente com de Queiroz e de Oliveira. O artigo foi preliminarmente aceito na conferência LSFA - 2015. Link do arxiv: http://arxiv.org/pdf/1504.04759v1.pdf (RAMOS; QUEIROZ; OLIVEIRA, 2015a)

o λ-cálculo não será o foco desse trabalho. A importância de introduzir esses conceitos básicos dessa teoria é que a teoria dos tipos se utiliza de diversos conceitos extremamente importantes do λ-cálculo. Um desses conceitos diretamente derivados são os axiomas da igualdade, que servem como o coração desse trabalho. Isso ficará mais claro quando o conceito de caminho computacional for formalmente introduzido.

Enquanto a máquina de Turing e o λ-cálculo nasceram com objetivos claramente computacionais, a teoria dos tipos nasceu como uma tentativa de formalização da matemática construtiva. Especificamente, uma tentativa baseada nos princípios do construtivismo de Errett Bishop. Devido à existência de diversas teorias dos tipos, inclusive uma oríunda do próprio λ-cálculo, é importante ressaltar que esse trabalho foca na teoria dos tipos construtiva, chamada de teoria **intuicionista** dos tipos, criada por Martin-Löf em 1971. Essa teoria dos tipos tenta substituir a estrutura fundamental que serve como base para a matemática. No caso, tenta substituir a ideia de conjuntos pela ideia de tipos. Todos os objetos matemáticos passam a ser imbuídos de um tipo, inclusive funções e até mesmo o conceito de prova matemática passa a ser associada a um tipo, como ficará claro através do tipo identidade. Apesar de ter surgido com o objetivo de servir como fundamento da matemática, a teoria dos tipos, devido ao construtivismo intríseco, rapidamente tornou-se uma das teorias que servem como fundamento da computação. Um exemplo disso é que existem certas linguagens funcionais que são diretamente baseadas nessa teoria, como Coq, Agda e Epigram.

Até recentemente, a teoria dos tipos era mais utilizada pelos conceitos computacionais. Porém, como ficou claro no capítulo passado, os problemas intrísecos do ZFC, juntamente com as novas descobertas envolvendo o tipo identidade, tornaram a comunidade matemática extremamente interessada na teoria dos tipos. Na próxima seção, a parte básica da teoria dos tipos será desenvolvida. Entender os conceitos da próxima seção será fundamental para entender os conceitos mais complexos que serão posteriormente desenvolvidos.

2.1 Conceitos básicos

Como não poderia deixar de ser, o conceito mais básico da teoria dos tipos é o conceito de tipo. O conceito de tipo deve ser entendido como um conceito fundamental no qual toda a teoria será baseada. Pense em um tipo como o equivalente ao conceito de conjunto na teoria dos conjuntos. Na verdade, o conceito de tipo é ainda mais poderoso que o conceito de conjuntos (UNIVALENT FOUNDATIONS PROGRAM, 2013). O motivo disso é que no ZFC, apesar de a teoria se basear no conceito de conjuntos, os objetos e os próprios axiomas são construidos a partir da utilização da lógica de primeira ordem. No ZFC, o conceito fundamental de conjunto e de proposição lógica são completamente distintos. Esse não é o caso da teoria dos tipos. Nessa teoria, não só o conceito de números, funções e demais objetos matemáticos são considerados tipos, mas também o proprío conceito de proposição também o é.

Seja A um tipo e a um termo do tipo A. É dito que A é habitado por a e denotado por

$a : A$ (lê-se o termo a tem tipo A). Na teoria dos tipos, esse julgamento é o mais básico. O outro formato de julgamento é $a = b : A$, i.e., a e b são elementos (intensionalmente) iguais do tipo A. Além disso, todo termo precisa ser acompanhado do respectivo tipo. Um termo a, sem o tipo acompanhado, não faz sentido. Dependendo da natureza do tipo A, é possível inferir interpretações diferentes (UNIVALENT FOUNDATIONS PROGRAM, 2013). Caso A seja uma proposição, $a : A$ pode ser entendido como a é uma prova que a proposição A é verdadeira. Neste caso, é dito que a é uma testemunha da veracidade de A. Caso A seja um conjunto, $a : A$ pode ser entendido como $a \in A$. Vale ressaltar que isso é apenas uma interpretação semântica, dizendo que a pode ser entendido como um elemento do conjunto A. Porém, como já foi dito, mesmo A sendo um tipo de um conjunto, não é permitido sintaticamente escrever que $a \in A$. Ainda é possível uma interpretação homotópica, quando A é um espaço topológico. Nesse caso, $a : A$ é interpretado como a sendo um ponto de A.

Com isso, conclui-se que apesar da existência de diversas interpretações, o único julgamento possível na teoria dos tipos é $a : A$. Essa é a ideia fundalmental que une, em torno da ideia de tipo, diversos conceitos da matemática, como caminhos topológicos e conjuntos, e da lógica, como o conceito de proposição.

2.1.1 Igualdade Definicional vs Proposicional

Na teoria dos tipos, tem dois tipos de igualdades distintas. A primeira, que é chamada de proposicional, é originada a partir do fato que a igualdade pode ser tratada como um tipo. A segunda é originada do fato que dois termos podem ser iguais simplesmente por definição, sem a necessidade de nenhum argumento mais desenvolvido que justifique a igualdade.

A igualdade proposicional é de extrema importância, já que é a responsável direta pelo tipo igualdade. A igualdade proposicional é originada da ideia que, dados termos $a, b : A$, é possível pensar no julgamento $a = b : A$. Tomando por base um julgamento como esse, é natural concebê-lo como premissa estabelecendo a igualdade (lógica, extensional) entre os elementos a e b de tipo A, e essa igualdade (lógica, extensional) dá origem a uma proposição, i.e. um tipo, conhecido como tipo identidade, geralmente representado por $Id_A(a,b)$. Usando a interpretação já mencionada, $Id_A(a,b)$ pode ser entendido como a proposição que estabelece a igualdade entre a e b. Nesse caso, diz-se que a é proposicionalmente igual a b. Caso exista alguma prova, i.e. uma sequência de aplicações de regras de reescrita, digamos p, que estabeleça que $a = b : A$, então $p : Id_A(a,b)$, ou seja, p é um termo constituído de uma composição dos identificadores das diversas reescritas utilizadas. Dessa forma, p será uma testemunha de $Id_A(a,b)$, estabelecendo a veracidade da proposição que a é proposicionalmente igual a b.

A igualdade definicional é mais simples que a proposicional. Nessa igualdade, não é necessário a existência de uma testemunha que indique que os objetos são iguais. Os objetos são iguais simplesmente por definição. O símbolo adotado para representar uma igualdade definicional é $\equiv$. Um exemplo simples é o seguinte (UNIVALENT FOUNDATIONS PROGRAM,

2013): Pode-se definir uma função f como $f(x) \equiv x^2$. Considere agora o caso de computar o valor de $f(3)$. De forma definicional, a única coisa que pode ser concluída é que $f(3) \equiv 3^2$. Não se pode afirmar que $f(3) \equiv 9$. O motivo disso, é que 3^2 não é, por definição, igual a 9. Para se chegar a esse resultado, é necessário obter uma prova p que estabeleça que $3^2 =_p 9 : \mathbb{N}$. Nesse caso, o que seria estabelecido seria a igualdade proposicional, já que se teria $p : Id_{\mathbb{N}}(3^2, 9)$. Para denotar que, dentro de um determinado tipo A, dois termos a e b são definicionalmente iguais, a notação utilizada é $a \equiv b : A$

Um exemplo ainda mais interessante que o dado anteriormente e que melhor demonstra a sutil diferença entre igualdade definicional e proposicional é dado a seguir:

Exemplo 2.1. Defina a adição da seguinte forma:

$$add : \begin{cases} a + 0 = a \\ a + s(b) = s(a + b) \end{cases}$$

Nesse exemplo, $s(x)$ é a função sucessor, definida como $s(x) \equiv x + 1$. A veracidade de alguns casos interessantes serão analisadas (HARPER, 2012):

- $2 + 2 \equiv 4 : \mathbb{N}$ (Verdadeiro): Diretamente da definição, $2 + 2 \equiv s(2 + 1) \equiv s(s(2 + 0)) \equiv s(s(2)) \equiv s(3) \equiv 4$.
- $0 + x \equiv x : \mathbb{N}$ (Falso): Aqui as sutilezas já começaram a aparecer. Como pela definição tem-se apenas que $x + 0 \equiv x$, é necessário provar que $0 + x = x$. Como é sabido, essa prova é facilmente obtenível. Mesmo assim, não involve apenas definições, a igualdade só se estabelece a nível proposicional.
- $s(x) \equiv (1 + x) : \mathbb{N}$ (Falso): Por definição, $s(x) \equiv x + 1$. Para mostrar que $1 + x = x + 1$, é necessário provar a comutatividade. Dessa forma, essa igualdade não se estabelece no âmbito definicional.
- $(x + y) \equiv (y + x) : \mathbb{N}$ (Falso): Pelo mesmo argumento do caso anterior.

Esse exemplo mostra de forma clara a limitação da igualdade definicional. Qualquer igualdade que necessite de alguma forma de evidência externa à definição, mesmo que seja algo simples como a comutividade dos naturais, não poderá ser estabelecida definicionalmente, tendo-se que recorrer a igualdade proposicional.

2.1.2 Família de tipos

Foi dito que um tipo pode representar proposições lógicas. Porém, não foi especificado como um tipo pode simular a ideia de predicado. Um exemplo simples é o seguinte: Dado um número natural x, decidir se x é par. Para criar um tipo que seria habitado se x é par, o próprio

tipo depende de x. Dessa forma, como para cada valor de x é criado um tipo novo, é dito que é definida uma familía de tipos indexadas pelos valores de x, que, nesse caso, são os naturais. Utiliza-se a notação $par(x)$, para indiciar que $par(x)$ é uma família de tipos indexada por x. Para exemplificar melhor, pense nessa família $par(x)$. Teria-se, então, $0 : par(0)$. No caso do tipo $par(1)$, como 1 é ímpar, a proposição 1 é par seria falsa, portanto o tipo $par(1)$ não seria habitado. Como é de se esperar, qualquer tipo da forma $par(2n+1)$ não será habitado, enquanto os tipos da forma $par(2n)$ serão habitados, $2n : par(2n)$.

Essa ideia de família indexada de tipos será básica para as diversas formulações das entidades da teoria dos tipos. Como pode-se ver, a família de tipos é equivalente ao conceito de predicado. A seguir, uma dedução natural com mais uma importante propriedade de família de tipos (HARPER, 2012):

$$\frac{m : A \qquad \begin{array}{c}[x:A]\\ B(x)\ \text{tipo}\end{array}}{[m/x]B(x)\ \text{tipo}}$$

A ideia é que dado uma família de tipos $B(x)$ indexadas pelo tipo A, dado um termo $m : A$, então obtém-se o tipo $[m/x]B(x) \equiv B(m)$. A outra propriedade importante é chamada de funcionalidade (HARPER, 2012):

$$\textit{funcionalidade} \qquad \frac{m \equiv n : A \qquad \begin{array}{c}[x:A]\\ B(x)\ \text{tipo}\end{array}}{[m/x]B(x) \equiv [n/x]B(x)\ \text{tipo}}$$

Essa propriedade é intuitiva. Dado uma familia de tipos indexadas pelo tipo A e dois elementos definicionalmente iguais $m \equiv n$, então $B(m) \equiv B(n)$. Um exemplo que e consequência direta da funcionalidade é que $par(2+2) \equiv par(1+3)$, já que $2+2 \equiv 1+3 : \mathbb{N}$.

2.1.3 O Tipo Função

O objetivo dessa subseção é introduzir os tipos que representam as funções. Uma função é construída a partir de um par de tipos A e B, originando o tipo $A \to B$. O tipo A é chamado de domínio e o tipo B de codomínio. No capítulo 1, uma função foi definida a partir do conceito de relação. A grande diferença entre a função da teoria dos tipos e a função de teoria dos conjuntos, é que, ao contrário da teoria dos conjuntos, a função da teoria dos tipos é um elemento primitivo da teoria (UNIVALENT FOUNDATIONS PROGRAM, 2013). Por ser um tipo primitivo, o conceito de função na teoria dos tipos é entendido a partir das formas que o tipo é construído e utilizado.

A parte mais básica do tipo função $f : A \to B$ é o fato que é possível aplicar um termo $a : A$ em f, obtendo-se um termo $b : B$. Nesse caso, a notação utilizada é a usual: $f(a) = b$

(algumas vezes, a notação $fa = b$ também é utilizada). Essa parte define a forma básica que uma função pode ser utilizada. Falta, porém, saber como uma função pode ser construída.

O tipo $A \to B$ pode ser construído a partir de duas formas básicas (UNIVALENT FOUNDATIONS PROGRAM, 2013). A primeira é diretamente a partir de uma igualdade definicional de $f(x)$. Nesse caso, $A \to B$ é construído a partir de uma $f(x) \equiv \theta$. As condições necessárias é que x seja do tipo A, ou seja, $x : A$ e que $\theta : B$. A segunda forma é definindo uma função usando a notação do λ-cálculo. Nesse caso, a função seria definidar a partir de uma abstração λx, obtendo-se $(\lambda(x : A).\theta) : A \to B$. Como o tipo de x já é inferido a partir do tipo da função, a parte $x : A$ geralmente é simplificada, obtendo-se $\lambda x.\theta : A \to B$. Usando essa segunda forma, uma aplicação é definida de forma equivalente a do λ-cálculo, com $(\lambda x.\theta)(a) \equiv [a/x]\theta$.

Como exemplo simples, pense em uma $f : \mathbb{N} \to \mathbb{N}$ em que recebe um $x : \mathbb{N}$ e retorna $x+2 : \mathbb{N}$. Uma forma de construir esse tipo seria uma igualdade definicional direta, obtendo-se $f(x) \equiv x+2 : \mathbb{N} \to \mathbb{N}$, com $f(a) \equiv a+2, a : \mathbb{N}$. A outra forma, usando o λ-cálculo, teria-se que $f = \lambda x.(x+2) : \mathbb{N} \to \mathbb{N}$. É claro, aplicando $a : \mathbb{N}$, o mesmo valor será obtido: $(\lambda x.(x+2))(a) \equiv [a/x](x+2) \equiv a+2$.

Outra parte extremamente importante de função como tipo é o conceito de função de múltiplas variáveis. Nos exemplos vistos até agora, a função recebia apenas uma variável. Porém, é possível que a função receba 2 ou mais varáveis. No caso de duas varaveis, teria-se, por exemplo, uma função $f(x, y : \mathbb{N}) \equiv \theta : N$. O tipo dessa função seria $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. O símbolo $\times$ não foi formalmente definido, mas é o produto cartesiano usual, o qual pode ser facilmente definido dentro da teoria dos tipos. Na notação de λ-cálculo, essa função poderia ser definida por $f = (\lambda xy.\theta) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Funções com mais de duas variáveis são definidas a partir de um processo totalmente análogo, apenas adicionando a quantidade necessaria de variáveis, além de adicionar $\times \mathbb{N}$ extras no produto cartesiano.

Por fim, existe um processo abundamente utilizado em funções de múltiplas varáiveis, chamado de currificação. A currificação é um processo de transformação de uma função a n-variáveis em n-funções que recebem apenas uma variável, mas que tem como retorno outra função (UNIVALENT FOUNDATIONS PROGRAM, 2013). Aqui será explicado o processo para uma função em duas variáveis, mas o mesmo processo pode ser analogamente utilizado para uma função de uma quantidade qualquer de variáveis. Dado uma função em duas variáveis, e que $f(a, b) \equiv c$, é possível transformar f em uma função que recebe apenas a e retorna uma outra função que recebe b, a qual, por sua vez, irá retornar c. Ao fazer a currificação, o tipo $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ é transformado no tipo equivalente $f : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$. Como o parênteses associa pela direita por definição, $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ pode ser escrito como $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$ sem qualquer ambiguidade.

Por ser excessivamente utilizado, é preciso entender bem a currificação. Por isso, considere um exemplo de uma função $f(x, y) \equiv x + y + 5$. Mais especificamente, considere a computação para $x = 3$ e $y = 5$. Sem currificação, teria-se que $f(3, 5) \equiv 3 + 5 + 5 \equiv 8 + 5 \equiv 13$. Com currificação, $f(x)$ passa a retornar uma $g : \mathbb{N} \to \mathbb{N}$ que recebe y. Nesse caso, teria-se que

$f(3) \equiv \lambda y.(3+y+5)$. Essa função, por sua vez, recebe o valor da entrada y. Com $y = 5$, teria-se $f(3)(5) \equiv 3+5+5 \equiv 13$. Como pode-se ver, apesar das duas abordagens ter tipos distintos e funcionar de formas diferentes, o resultado final é exatamente o mesmo. Dessa forma, qualquer função em múltiplas variáveis é equivalente a uma de apenas uma variável após passar pelo processo de currificação.

2.1.4 Função de Tipo Dependente (Π)

A função de tipo dependente surge a partir da necessidade de expressar um tipo que é impossível de ser construído a partir do tipo função da subseção anterior. Esse é o caso em que a saída é um tipo que faz parte de uma família de tipos indexada pelo tipo da entrada. Para melhor explicar, pense em um tipo $Seq(x)$, indexado pelos naturais. Um termo de um tipo $Seq(x)$ é a seqüência de naturais de 0 até x. Pense no problema de construir uma função que tenha como entrada $n : \mathbb{N}$ e saída $\{0...n\} : Seq(n)$. Como o tipo da saída é variável e depende da entrada, o conceito usual de tipo função não consegue definir uma função para esse caso (já que na função o tipo da saída é fixo). Para indicar que o tipo é uma função dependente, é utilizada a notação $\Pi_{(x:A)}B(x)$, com x sendo a entrada e $B(x)$ uma família de tipos indexadas pelo tipo A. No exemplo de $Seq(x)$, teria-se a função $\Pi_{(x:\mathbb{N})}Seq(x)$. A seguir serão mostradas regras de inferência que indicam como o tipo de função dependente é construído e utilizado (HARPER, 2012):

$$\frac{A \text{ tipo} \qquad \begin{matrix}[x:A]\\ B(x) \text{ tipo}\end{matrix}}{\Pi_{(x:A)}B(x) \text{ tipo}}\ (\Pi - F) \qquad \frac{\begin{matrix}[x:A]\\ m(x) : B(x)\end{matrix}}{\lambda(x:A).m(x) : \Pi_{(x:A)}B(x)}\ (\Pi - I)$$

As duas deduções foram colocadas juntas, já que uma complementa a outra. No caso de Formação do Produto Dependente ($\Pi - F$), é o que tinha sido discutido anteriormente. Dado um tipo A e uma família $B(x)$ indexada por esse tipo, é formado o tipo $\Pi_{(x:A)}B(x)$. O caso de Introdução do Produto Dependente ($\Pi - I$) tambem é simples. Dado um termo $m(x) : B(x)$, com $x : A$, ou seja, um termo de uma família indexada por A, é possivel obter uma função dependente que recebe o tipo x e retorna o tipo $m(x)$. Agora, a regra de eliminação:

$$\frac{n : A \qquad m : \Pi_{(x:A)}B(x)}{m.n : [n/x]B(x)}\ (\Pi - E)$$

A regra de Eliminação do Produto Dependente ($\Pi - E$) recebe esse nome por ser a forma em que o símbolo do produto dependente é eliminado. Olhando para a derivação acima, percebe-se que a eliminação é basicamente a aplicação de um elemento n na função dependente. O próprio tipo resultante da aplicação é muito parecido com o da aplicação na forma não-dependente, sendo

a única diferença que o tipo depende da entrada. Apesar dessa regra definir o tipo resultante da aplicação, ela não define como é feita a aplicação em uma função dependente. Por isso, é necessário mais uma regra, a regra de computação:

$$\frac{\begin{array}{c}[x:A]\\ n:A \qquad m(x):B(x)\end{array}}{(\lambda(x:A).m).n = [n/x]m(x) : [n/x]B(x)}\ (\Pi - C)$$

Olhando atentamente, a regra de Computação do Produto Dependente $(\Pi - C)$ estabelece a aplicação de uma forma exatamente igual a utilizada em funções não-dependentes. A única diferença é que o termo e o tipo resultante dependem da entrada x. Com isso, todas as regras necessárias para construir e computar o tipo função depedente foram mostradas.

2.1.5 Tipo par dependente (Σ)

Ao trabalhar com funções, é usual trabalhar com pares ordenados (a,b) em que a é a entrada e b é a imagem ao aplicar a. Porém, o que aconteceria ao usar o mesmo tipo de raciocínio para uma função dependente? O tipo da imagem dependeria do valor da entrada, portanto o tipo do par ordenado teria que considerar esse fato. Nesses casos, é necessário utilizar um tipo chamado de par dependente, em que o tipo do segundo elemento do par ordenado depende do valor do primeiro elemento. A notação utilizada é $\Sigma_{(x:A)}B(x)$ e o tipo possui as regras (HARPER, 2012):

$$\frac{\begin{array}{c}[x:A]\\ A \text{ tipo} \qquad B(x) \text{ tipo}\end{array}}{\Sigma_{(x:A)}B(x) \text{ tipo}}\ (\Sigma - F)$$

$$\frac{m:A \qquad n:B(m)}{\langle m,n\rangle : \Sigma_{(x:A)}B(x)}\ (\Sigma - I)$$

A Formação da Soma Dependente $(\Sigma - F)$ é similar a $\Pi - F$. Para a Introdução da Soma Dependente $(\Sigma - I)$, um elemento $m:A$ e um elemento $n:B(m)$ são o suficiente para introduzir um par dependente $\langle m,n\rangle : \Sigma_{(x:A)}B(x)$. Agora, as duas regras de eliminação:

$$\frac{m:\Sigma_{(x:A)}B(x)}{\pi_1(m):A}\ (\Sigma - E_1) \qquad \frac{m:\Sigma_{(x:A)}B(x)}{\pi_2(m):[\pi_1(m)/x]B(x)}\ (\Sigma - E_2)$$

Ao contrário do $\Pi - E$, o par dependente tem dois tipos de eliminação. A justificativa para isso é direta. Como o par tem dois termos, é possível eliminar Σ de forma a obter o primeiro ou o segundo termo. A Primeira Eliminação da Soma Dependente $(\Sigma - E_1)$ é óbvia. A partir

de um par ordenado, é possivel extrair o primeiro elemento, o qual tem o tipo nao-dependente. A Segunda Eliminação da Soma Dependente ($\Sigma - E_2$) é similar a primeira, mas o tipo obtido depende do primeiro valor do par ordenado. Com isso, basta saber como computar com um par ordenado:

$$\pi_1(\langle m,n\rangle) = m \text{ e } \pi_2(\langle m,n\rangle) = n.$$

Como pode-se ver, os operadores π_1 e π_2 extraem a informação necessária do par dependente, obtendo-se m e n respectivamente.

Não obstante, desde o artigo original de W. Howard (HOWARD, 1980) sobre o paradigma *formulae-as-types*, essas regras de eliminação do par dependente podem levar a problemas com relação à correspondência com as regras do quantificador existencial devido à violação da condição de 'esconder' a testemunha de um enunciado existencial (QUEIROZ; GABBAY, 1995). Assim, uma apresentação da eliminação do Σ que se revela mais apropriada à correspondência com o quantificador existencial tem a forma:

$$\frac{\begin{array}{cc} & [t:A,\ f(t):B(t)] \\ n:\Sigma_{(x:A)}B(x) & h(t,f):C \end{array}}{E(n,\lambda t.\lambda f.h(t,f)):C}\ (\Sigma - E)$$

2.1.6 Os Naturais

Foi dito que um dos objetivos da teoria dos tipos é servir como fundamento para a matemática. Uma teoria que se propõe a ser fundamento da matemática deve ser capaz, no mínimo, de definir formalmente a noção de números naturais. Esse é o caso, por exemplo, da teoria dos conjuntos. Assim como na teoria dos conjuntos, a teoria dos tipos é completamente capaz de formalizar os números naturais. Nessa subseção, o objetivo será mostrar como os naturais são construídos e algumas propriedades básicas. Para começar, seguem as deduções naturais das construções básicas (HARPER, 2012):

$$\frac{}{0:\mathbb{N}}\ (\mathbb{N}-I_0) \qquad \frac{m:\mathbb{N}}{suc(m):\mathbb{N}}\ (\mathbb{N}-I_s)$$

Os naturais são construídos de forma extremamente simples, a partir das duas regras acima. A Introdução do Zero ($\mathbb{N}-I_0$) deve ser entendida como a base da construção. Diz que existe um número 0 que é um termo do tipo $\mathbb{N}$. A Indução do Sucessor ($\mathbb{N}-I_s$) deve ser entendida como o passo indutivo. Dado qualquer m natural, é possível afirmar que o sucessor $suc(m)$ também é natural. Se, a partir de 0, a segunda regra for aplicada, então todos os naturais serão obtidos como conseqüência. As regras mais complicadas, porém, são as que definem as computações que podem ser feitas a partir dos naturais (HARPER, 2012):

$$\frac{m : \mathbb{N} \qquad C \text{ tipo} \qquad N_0 : C \qquad \begin{matrix}[x : c] \\ N_s : C\end{matrix}}{rec(m, N_0, \lambda x.N_s) : C}$$

Para alguém não familizado com ideia de recursão primitiva, a regra de inferência acima pode ser bastante confusa. A ideia é que as funções básicas envolvendo naturais, como adição e multiplicação, podem ser definidas a partir de um recursor, representado por *rec* na dedução. O recursor funciona a partir de um caso base N_0, que indica a computação a se fazer no caso de 0 iterações. Outra informação necessária é que dado uma iteração x, é necessário saber como computar a próxima iteração, a qual é representada por N_s. Com isso, o recursor é capaz de definir exatamente como funciona a função. Por exemplo, usando o recursor, é possível definir a adição. Dados $\lambda x : \mathbb{N}$ e $\lambda y : \mathbb{N}$, no caso de 0 iteraçoes, basta pegar o valor de x e no caso de z iterações, a próxima iteração pode ser definida como $\lambda z.suc(z)$. Dessa forma, a adição é representada por $rec(y, x, \lambda z.suc(z))$. Como o tipo C é um tipo não-dependente, o recursor é conhecido como forma não-dependente do recursor dos naturais. Existe uma forma dependente (HARPER, 2012):

$$\frac{m : \mathbb{N} \qquad \begin{matrix}[x : \mathbb{N}] \\ C(x) \text{ tipo}\end{matrix} \qquad N_0 : [0/x]C(x) \qquad \begin{matrix}[x : \mathbb{N}, y : C(x)] \\ N_s : [suc(x)/x]C(x)\end{matrix}}{rec(m, N_0, (x, \lambda y.Ns)) : [m/x]C(x)}$$

À primeira vista, a forma dependente parece mais complicada que a não-dependente. Porém, ao olhar mais atentamente, logo percebe-se que a dedução natural acima é o princípio da indução dos naturais. Basicamente, dado uma prova do caso base $N_0 : C(0)$ e o passo indutivo, ou seja, se uma prova $y : C(x)$ implica a existência de uma prova $N_s : C(x+1)$, então tem-se uma prova $C(m)$ representada pelo recursor para um $m : \mathbb{N}$ qualquer. Dessa forma, a prova é válida para todos os naturais. Com isso, o objetivo dessa subseção de mostrar a construção e os princípios básicos dos naturais a partir da teoria dos tipos está concluído.

2.1.7 Informações Adicionais

Como foi visto anteriormente, um tipo pode ser interpretado de diversas formas, utilizando o ponto de vista da lógica, teoria dos conjuntos e homotopia. As principais interpretações são resumidas na tabela a seguir:

Como visto na tabela acima, a teoria dos tipos é capaz de representar os mais diversos objetos da matemática e da lógica, portanto, mostrar cada construção detalhadamente seria completamente inviável. Porém, as subseções anteriores cumpriram o papel de mostrar as construções mais importantes e que podem causar mais problemas para um leitor não familiarizado com a teoria. Entretanto, é de extrema importância destacar que outros conceitos importantes,

Quadro 2.1: As múltiplas interpretações de diversos tipos

Tipos	**Lógica**	**Conjuntos**	**Homotopia**
A	proposição	conjunto	espaço
$a : A$	prova	elemento	ponto
$B(x)$	predicado	família de conjuntos	fibração
$b(x) : B(x)$	prova condicional	família de elementos	secção
0,1	$\bot, \top$	$\varnothing, \{\varnothing\}$	$\varnothing, *$
$A + B$	$A \vee B$	união disjunta	coproduto
$A \times B$	$A \wedge B$	produto cartesiano	espaço produto
$A \to B$	$A \Rightarrow B$	exponenciação	espaço função
$\Sigma_{x:A} B(x)$	$\exists_{(x:A)} B(x)$	soma disjunta	espaço total
$\Pi_{x:A} B(x)$	$\forall_{(x:A)} B(x)$	produto	espaço das secções
Id_a	igualdade $=$	$\{(x,x) \mid x \in A\}$	espaço de caminhos A^I

Fonte: Livro Homotopy Type Theory (UNIVALENT FOUNDATIONS PROGRAM, 2013).

como o produto cartesiano ($\times$), a soma disjunta ($+$) e o conceito de booleanos $(0,1)$. Essas construções podem e são completamente formalizáveis a partir da teoria dos tipos. Além disso, apesar de poder ser considerado como um conceito básico, a formalização do tipo identidade foi omitida nessa subseção. O objetivo disso é que, dado a importância desse tipo, a próxima seção será inteiramente dedicada ao tipo identidade.

2.2 Tipo identidade

Após passar rapidamente pelos principais conceitos da teoria dos tipos, esta seção finalmente tem como objetivo uma exposição detalhada do tipo identidade. O objetivo inicial será mostrar a abordagem atual, a qual é complexa e não intuitiva. Logo depois, será feita uma discussão com as diferenças entre duas teoria dos tipos distintas: a intensional e a extensional. Será visto que a extensionalidade ou a intensionalidade depende diretamente de como o tipo identidade é formulado. Após ler essa seção, ficará claro para o leitor a necessidade da criação de uma nova abordagem simples e intuitiva que defina satisfatoriamente o tipo identidade.

Antes de entrar em detalhes mais complexos sobre o tipo identidade, é necessário entender a ideia intuitiva deste tipo. Como foi visto, a teoria dos tipos tem uma diferença clara entre igualdade proposicional e definicional. Enquanto a igualdade definicional se justifica por si só, a igualdade proposicional precisa de uma evidência que sustente a igualdade. Em outras palavras, a igualdade proposicional necessita de uma prova que estabeleça $a = b : A$, com $a, b : A$. Como, na teoria dos tipos, proposições são tipos, então $a = b : A$ pode ser representado por um tipo, o qual é chamado de tipo identidade e representado por $Id_A(a,b)$. Dessa forma, os termos $p : Id_A(a,b)$ são as testemunhas que estabelecem que a é proposicionalmente igual a b. Dessa forma, o tipo identidade faz completo sentido do ponto de vista intuitivo. Entretanto, como foi feito para os conceitos fundamentais da seção anterior, como função e pares dependentes, o tipo identidade precisa ser formalmente introduzido e leis que definem exatamente como pode ser utilizado precisam ser definidas. De forma semelhante, essas construções e regras são propostas

a partir de regras de inferência. Como será visto a seguir, são exatamente essas deduções naturais que são extremamente não-intuitivas, tornando um tipo que deveria ser intuitivo, como o tipo identidade, em uma entidade bastante complexa de ser utilizada.

2.2.1 Abordagem Atual do Tipo Identidade

Assim como foi feito na seção anterior, a construção será feita a partir das regras de inferência. Seguem as deduções que constroem o tipo (QUEIROZ; OLIVEIRA, 2014a):

$$\frac{A \text{ tipo} \qquad a : A \qquad b : A}{Id_A^{int}(a,b) \text{ tipo}}\, Id^{int} - F \qquad \frac{a : A}{r(a) : Id_A^{int}(a,a)}\, Id^{int} - I$$

Primeiramente, é importante esclarecer o significado do *int* presente em Id^{int}. O *int* indica que essa é a versão intensional do tipo identidade. O significado das versões intencionais e extensionais e a diferença entre elas será o tópico da proxima subseção. O importante agora é esclarecer que a versão utilizada na teoria homotópica dos tipos e que está sendo definido agora é a versão intensional. Daqui em diante, se *int* ou *ext* não for especificado, considere que está sendo utilizada a versão intensional.Com essa parte esclarecida, pode-se começar a analisar as regras mostradas acima. A Formação do Tipo Identidade Intencional ($Id^{int} - F$) não acarreta surpresas. Dado dois termos $a, b : A$, é possível criar um tipo que será habitado caso exista uma evidência que sirva de testemunha do fato que os dois termos são proposicionalmente iguais. Nesse caso, independente de o tipo ser habitado ou não, sempre será possível cria-lo, obtendo-se $Id_A(a,b)$.

A regra de Introdução do Tipo Identidade Intencional ($Id^{int} - I$) é o que causa todo o estranhamento e a não-intuitividade do tipo identidade. $r(a)$ representa uma prova por reflexividade. Isso é, dado um termo $a : A$, $r(a)$ é a prova que $a = a : A$. Naturalmente a reflexividade é um princípio básico de qualquer teoria que envolve igualdade. Portanto, é natural pensar que dado um tipo $a : A$, sempre existirá uma prova $r(a) : Id_A(a,a)$. Então, qual seria a parte não intuitiva? É o fato que a reflexividade é a única forma de introduzir o tipo identidade. Isso é equivalente a dizer que qualquer que seja a prova da igualdade, é sempre possível obtê-la a partir da prova da reflexividade. Pense nos naturais. Os naturais tem duas regras básicas, a existência do 0 e que para um número n existe o sucessor de n, $suc(n)$. Ora, construir um número m a partir dessas regras é extremamente intuitivo. Basta partir do 0 e aplicar a regra do sucessor a quantidade necessária de vezes, obtendo-se m eventualmente. A mesma intuitividade não é obtida a partir da regra de introdução do tipo identidade. Como é possível obter possíveis provas complexas de um $Id_A(a,b)$ a partir apenas da prova da reflexividade? Ou até provas simples, como a transitividade e a simetria da igualdade? De fato, geralmente as teorias da igualdade consideram a própria transitividade e simetria como axiomas adicionais, junto com a reflexividade. É verdade que, como será visto a seguir, é possível definir um recursor J que transforma a reflexividade em qualquer prova genérica. E a partir desse recursor, provar a

transitividade e a simetria. Entretanto, a forma que esse recursor atua é extremamente obscura e não intuitiva. Segue a regra de inferência que introduz J (QUEIROZ; OLIVEIRA, 2014a):

$$\frac{a:A \qquad b:A \qquad c:Id_A^{int}(a,b) \qquad \begin{matrix}[x:A]\\ d(x):C(x,x,r(x))\end{matrix} \qquad \begin{matrix}[x:A,y:A,z:Id_A(x,y)]\\ C(x,y,z)\text{ tipo}\end{matrix}}{J(c,d):C(a,b,c)}\ Id-E$$

Como pode-se ver, a regra de Eliminação do Tipo Identidade ($Id-E$) que introduz o recursor J e elimina o tipo identidade é extensa. Porém, é possível entender o objetivo do recursor J. Basicamente $J(c,d):C(a,b,c)$ funciona construindo c a partir de uma prova reflexiva d. Nesse caso, como em qualquer recursor, c é construído apos múltiplas iterações que se iniciam na base d. O grande problema, porém, é que o funcionamento de J não é bem explicado pela dedução natural. O que acontece em cada iteração não é explicado. É verdade que J obtém d a partir de c, mas não é explicado como isso acontece. É possível justificar o funcionamento de J a partir dos conceitos de teoria das categorias. Entretanto, os conceitos utilizados são complexos e os argumentos demasiadamente complicados. Dado que J é extensivamente utilizado para construir termos em uma grande quantidade de provas da teoria dos tipos, a obscuridade e complexidade do funcionamento interno de J torna-se uma grande desvantagem. Juntando J com a não-intuitividade de obter qualquer prova a partir apenas da reflexividade, obtém-se um tipo identidade extremamente complexo, tornando-se uma das entidades básicas mais complicadas da teoria dos tipos. A seguir, a última regra envolvendo o tipo identidade (QUEIROZ; OLIVEIRA, 2014a):

$$\frac{a:A \qquad \begin{matrix}[x:A]\\ d(x):C(x,x,r(x))\end{matrix} \qquad \begin{matrix}[x:A,y:A,z:Id_A(x,y)]\\ C(x,y,z)\text{ tipo}\end{matrix}}{J(r(a),d(x))=d(a/x):C(a,a,r(a))}\ Id-Eq$$

A regra acima foi mostrada mais por questões de completude. Conhecida como Igualdade do Tipo Identidade ($Id-Eq$), é uma regra intuitiva. Dado uma prova da reflexividade $d(x)$, construir a prova da reflexividade $r(a)$ a partir de J é igual a simplesmente colocar a no lugar de x, obtendo-se $d(a):C(a,a,r(a))$. Com isso, é finalizada toda a parte necessária sobre a abordagem intensional do tipo identidade.

2.2.2 Extensionalidade vs Intensionalidade

Na teoria dos conjuntos, foi visto que as funções sao tratadas de forma extensional. As funções são consideradas como pares de entrada e saída. A forma com que os pares sao obtidos é desconsiderada. Isso não é o que acontece na formulação do tipo identidade abordada na subseção anterior. Nesse caso, o tipo identidade deixa claro que a forma que a igualdade é obtida é de grande importância. Por isso, é dito que essa abordagem é intensional. Ao formular

a teoria dos tipos, Martin-Löf propôs uma abordagem intensional (MARTIN-LÖF, 1998) e uma extensional (MARTIN-LÖF, 1982; MARTIN-Löf, 1984). Como a versão intensional já foi abordada, essa subseção tem o objetivo de mostrar a versão extensional. Para isso, serão mostradas a construção e utilização do tipo identidade extensional, denotado por Id^{ext}. Seguem as deduções (QUEIROZ; OLIVEIRA, 2011):

$$\frac{A \text{ tipo} \qquad a:A \qquad b:A}{Id^{ext}_A(a,b) \text{ tipo}}\, Id^{ext}-F \qquad \frac{a=b:A}{r:Id^{ext}_A(a,b)}\, Id^{ext}-I$$

Em particular, nenhum comentário é necessário para a Formação do Tipo Identidade Extensional ($Id^{ext}-F$), já que é exatamente igual ao do tipo intensional. A Introdução do Tipo Identidade Extensional ($Id^{ext}-I$) também é simples. Se é fato que $a=b:A$, então é porque existe uma prova r que justifica a igualdade. Naturalmente, r tem tipo $Id^{ext}_A(a,b)$. A seguir, a eliminação e a igualdade (QUEIROZ; OLIVEIRA, 2011):

$$\frac{c:Id^{ext}_A(a,b)}{a=b:A}\, Id^{ext}-E \qquad \frac{c:Id^{ext}_A(a,b)}{c=r:Id^{ext}_A(a,b)}\, Id^{ext}-Eq$$

Talvez a primeira característica a ser reparada é o quão simples é a Eliminação do Tipo Identidade Extensional ($Id^{ext}-E$) quando comparada com a versão intensional. Não existe a necessidade de estruturas complicadas como o recursor J. Na versão extensional, dado um elemento $c:Id^{ext}_A(a,b)$, a única informação relevante a ser extraida é que $a=b:C$. Nesse ponto, a informação que c é o responsável por estabelecer a prova passa a ser irrelevante. Como pode ser visto, a própria existência de c é completamente excluída. Isso acontece porque a versão extensional não se preocupa em guardar a informação de como uma determinada prova da igualdade foi alcançada. A regra de Igualdade do Tipo Identidade Extensional ($Id^{ext}-Eq$) também demonstra claramente esse fato. A igualdade mostra que uma prova c é igual a uma prova r qualquer. Fazendo um paralelo com a teoria dos conjuntos, é como se c e r fossem funções que possuem os mesmos pares ordenados. Dessa forma, por mais diferentes que possam ser as partes internas das provas c e r, elas possuem a mesma saída, que é o estabelecimento do fato que $a=b:A$.

Caso a computação fosse baseada na teoria dos tipos extensional, propor uma nova formulação para o tipo identidade seria contraprodutivo. Isso se deve ao fato que as regras da versão extensional são bastante simples e fáceis de utilizar. Contudo, já foi dito que a computação é naturalmente intensional. Além disso, até a própria matemática é beneficiada pela versão intensional, já que é a versão que deu origem à interpretação homotópica. Dado a complexidade da formulação do tipo identidade, o objetivo de propor uma nova abordagem é propor uma formulação mais natural, cujas regras são mais fáceis de utilizar e entender. É nesse contexto que surge o conceito de caminhos computacionais, entidade que será estudada na próxima seção.

2.3 Caminhos Computacionais

Essa seção tem o objetivo de introduzir a entidade mais importante desse trabalho: Os caminhos computacionais. Essa entidade é baseada na ideia que é possível definir formalmente quando um objeto computacional a é igual a um objeto computacional b. É possível definir um conjunto de axiomas e operações que tornam possível transformar a em b. Essa transformação acontece ao a partir de a se aplicar em sequência as operações desse conjunto de axiomas até obter-se b. Como cada operação transforma um termo em outro, elas são também conhecidas como reescritas. É por isso que caminhos computacionais também podem ser chamados de seqûencia de reescritas. Contudo, antes de definir formalmente o conceito de caminho computacional e mostrar os axiomas de reescrita (que são os axiomas de igualdade), é importante mostrar a teoria em que os axiomas serão baseados, a qual é conhecida como Teoria da Beta-Eta Igualdade ($\lambda\beta\eta$).

2.3.1 Igualdade $\lambda\beta\eta$

A igualdade $\lambda\beta\eta$ é uma teoria axiomática de igualdade formal do λ-cálulo que estabelece a igualdade $\beta\eta$ entre dois termos do λ-cálulo. Porém, para entender satisfatoriamente esse conjunto de axiomas é necessário enteder alguns conceitos básicos do λ-cálculo:

Definição 2.1 (Variáveis livres e ligadas (HINDLEY; SELDIN, 2008))**.** *Uma varíavel x em um termo P pode ser:*

- *Ligada se está dentro do escopo de um λ x em P,*
- *Ligada e ligante se é o x em λx,*
- *Livre caso nenhum dos dois casos se aplicam.*

O conjuntode todas as variáveis livres de P é detonado por $FV(P)$.

Definição 2.2 (Mudança de variável ligada e congruência (HINDLEY; SELDIN, 2008))**.** *Seja P um termo que tem uma ocorrência $\lambda x.M$ e $y \notin FV(M)$. Então, a troca de $\lambda x.M$ por $\lambda y.[y/x]M$ é chamado de troca de variável ligada. Se for possível transformar o termo P em um termo Q a partir de finitas (ou até mesmo zero) séries de mudanças de variáveis ligadas, então é dito que P é congruente a Q ou P α-converte a Q e é denotado por $P \equiv_\alpha Q$.*

Definição 2.3 (β-contração e β-redução (HINDLEY; SELDIN, 2008))**.** *Um termo da forma $(\lambda x.M)N$ é chamado de $\beta - redex$ e a forma correspondente $[N/x]M$ de $\beta - contractum$. Dado um termo P e uma $\beta - redex$ $(\lambda x.M)N$, é possível trocar a redex por $[N/x]M$, obtendo-se um termo P'. É dito que a redex de P foi contraída e que P β-contrai para P' e é denotado por $P \rhd_{1\beta} P'$. Se P pode ser transformado em Q através de finitas (ou até mesmo 0) β-contrações ou mudanças de variáveis ligadas, então é dito que P beta-reduz para Q e é denotado por $P \rhd_\beta Q$.*

Definição 2.4 (β-igualdade (HINDLEY; SELDIN, 2008)). *É dito que P é β-igual a Q (denotado por $P =_\beta Q$ se Q pode ser obtido a partir de P por uma quantidade finita (ou talvez 0) de séries de β-contrações e β-contrações reversas e mudanças de variáveis ligadas. Em outras palavras, $P =_\beta Q$ sse existe $P_0, ..., P_n (n > 0)$ tal que:*

$$(\forall i \leq n-1)(P_i \rhd_{1\beta} P_{i+1} \text{ ou } P_{i+1} \rhd_{1\beta} P_i \text{ ou } P_i \equiv_\alpha P_{i+1})$$
$$P_0 \equiv P, \quad P_n \equiv Q$$

A formalização axiomática da β-igualdade junto com uma regra conhecida como η, cujo funcionamento ficará claro nos axiomas, gera a teoria da $\lambda\beta\eta$-igualdade:

Definição 2.5 (Teoria da $\lambda\beta\eta$-igualdade (HINDLEY; SELDIN, 2008)). *A teoria é composta pelos seguintes axiomas:*

- $(\alpha)\ \lambda x.M = \lambda y.[y/x]M \quad$ *se* $y \notin FV(M)$;
- $(\beta)\ (\lambda x.M)N = [N/x]M$;
- $(\rho)\ M = M$;
- $(\eta)\ (\lambda x.Mx) = M \quad (x \notin FV(M))$.

E as seguintes regras de inferência:

$$(\mu) \quad \frac{M = M'}{NM = NM'} \qquad (\tau)\ \frac{M = N \qquad N = P}{M = P}$$

$$(\nu) \quad \frac{M = M'}{MN = M'N} \qquad (\sigma)\ \frac{M = N}{N = M}$$

$$(\xi) \quad \frac{M = M'}{\lambda x.M = \lambda x.M'}$$

$$(\zeta) \quad \frac{Mx = Nx}{M = N}$$

Se a equação $M = N$ é provável em $\lambda\beta\eta$, é dito que $\lambda\beta\eta \vdash M = N$.

Com esse conjunto de regras e axiomas é possível, então, definir $\lambda\beta\eta$-igualdade:

Definição 2.6 ($\lambda\beta\eta$-igualdade (QUEIROZ; OLIVEIRA, 2011)). *A relação de igualdade definida por $\lambda\beta\eta$-igualdade é chamada de $=_{\beta\eta}$. Em outras palavras:*

$$M =_{\beta\eta} N \Leftrightarrow \lambda\beta\eta \vdash M = N.$$

Na próxima subseção será visto como os caminhos computacionais podem ser definidos a partir de uma teoria equivalente à teoria da $\lambda\beta\eta$-igualdade.

2.3.2 Formalização do Caminho Computacional

Já foi exposto que o conceito de caminhos computacionais parte da ideia de definir formalmente quando dois objetos computacionais são iguais. É exatamente o que a teoria da $\lambda\beta\eta$-igualdade faz. Decide quando dois termos do λ-cálculo são iguais (mais especificamente, $\beta\eta$-iguais). Porém, como os objetos computacionais são termos da teoria dos tipos, não é possível utilizar diretamente a teoria anterior, já que é uma teoria do λ-cálculo. Felizmente, é possível traduzir a teoria $\lambda\beta\eta$-igualdade para uma teoria de igualdade da teoria dos tipos. Para o tipo Π, os seguintes axiomas são obtidos:

Definição 2.7 (Teoria da igualdade da teoria dos tipos para Π (QUEIROZ; GABBAY, 1994; QUEIROZ; OLIVEIRA, 2014b, 2011))**.** *A teoria da igualdade da teoria dos tipos para o tipo* Π *é composta dos seguintes axiomas:*

$$(\beta)\quad \frac{\begin{array}{cc}[x:A]\\ N:A & M:B\end{array}}{(\lambda x.M)N = M[N/x] : B} \qquad (\xi)\quad \frac{[x:A]}{\begin{array}{c}M = M' : B\\ \hline \lambda x.M = \lambda x.M' : (\Pi x : A)B\end{array}}$$

$$(\rho)\quad \frac{M:A}{M = M : A} \qquad (\mu)\quad \frac{M = M' : A \qquad N : (\Pi x : A)B}{NM = NM' : B}$$

$$(\sigma)\quad \frac{M = N : A}{N = M : A} \qquad (\nu)\quad \frac{N : A \qquad M = M' : (\Pi x : A)B}{MN = M'N : B}$$

$$(\tau)\quad \frac{M = N : A \qquad N = P : A}{M = P : A}$$

$$(\eta)\quad \frac{M : (\Pi x : A)B}{(\lambda x.Mx) = M : (\Pi x : A)B}\ (x \notin FV(M))$$

Olhando para os axiomas acima, pode-se ver claramente que cada axioma e regra de inferência da $\lambda\beta\eta$-igualdade foram diretamente traduzidos para um axioma equivalente formulado utilizando a teoria dos tipos. Essa teoria torna possível definir o conceito de caminho computacional:

Definição 2.8 (Caminho computacional (QUEIROZ; OLIVEIRA, 2011))**.** *Seja a e b elementos de um tipo A. Um caminho computacional (ou sequência de reescritas) s de a até b é uma sequência de igualdades definicionais (cada igualdade definicional é uma aplicação de um dos axiomas da teoria de igualdade da teoria dos tipos ou é uma mudança de variável ligada) começando em a e terminando em b. É dito que $a =_s b : A$.*

A definição já deixa claro a vantagem de se trabalhar com a abordagem utilizando caminhos computacionais. Cada sequência de reescrita é a aplicação de um dos axiomas acima. Os axiomas contém regras bastante comuns em qualquer teoria da igualdade, como axiomas que representam a reflexividade (ρ), transitividade (τ) e simetria (σ). Além desses axiomas clássicos, os axiomas restantes são derivados de regras simples de uma teoria amplamente estudada, que é a teoria do λ-cálculo. Por isso, o caminho computacional torna a ideia de

igualdade completamente intuitiva, sendo um ótimo candidato a formalizar o tipo identidade. De fato, esta formalização será feita na próxima subseção. Não obstante, a definição anterior da teoria da igualdade, enriquecida com os identificadores básicos para cada tipo de igualdade que serão utilizados como base para a construção dos caminhos computacionais, fica da seguinte forma:

Definição 2.9 (Teoria da igualdade da teoria dos tipos para Π com identificadores para igualdades (QUEIROZ; OLIVEIRA, 2011))**.** *A teoria da igualdade da teoria dos tipos para o tipo* Π *é composta dos seguintes axiomas:*

$$(\beta)\quad \frac{\begin{array}{cc}[x:A]\\ N:A \qquad M:B\end{array}}{(\lambda x.M)N =_{\beta} M[N/x] : B} \qquad (\xi)\quad \frac{[x:A]}{\lambda x.M =_{\xi(s)} \lambda x.M' : (\Pi x:A)B}$$

$$(\rho)\quad \frac{M:A}{M =_{\rho} M : A} \qquad (\mu)\quad \frac{M =_s M' : A \qquad N : (\Pi x:A)B}{NM =_{\mu(s)} NM' : B}$$

$$(\sigma)\quad \frac{M =_s N : A}{N =_{\sigma(s)} M : A} \qquad (\nu)\quad \frac{N : A \qquad M =_s M' : (\Pi x:A)B}{MN =_{\nu(s)} M'N : B}$$

$$(\tau)\quad \frac{M =_s N : A \qquad N =_t P : A}{M =_{\tau(s,t)} P : A}$$

$$(\eta)\quad \frac{M: (\Pi x:A)B}{(\lambda x.Mx) =_{\eta} M: (\Pi x:A)B}\ (x \notin FV(M))$$

A seguir, um exemplo que melhor demonstra a utilização dos caminhos computacionais na prática:

Exemplo 2.2. O termo $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$ é proposicionalmente igual a $N \equiv zv$ devido a sequência:
$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, \quad (\lambda x.(\lambda y.yx)z)v, \quad (\lambda y.yv)z, \quad zv$
Transformar essa sequência em um caminho resulta em:
O primeiro termo é igual ao segundo devido a:
$\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v)$
O segundo é igual ao terceiro devido a:
$\beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)$
Portanto, o primeiro é igual ao terceiro devido a:
$\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z))$
O terceiro é igual ao quarto devido a:
$\beta((\lambda y.yv)z, zv)$
Finalmente, o primeiro é igual ao último devido a:
$\tau(\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)), \beta((\lambda y.yv)z, zv))).$
Portanto, este último caminho estabelece a igualdade proposicional entre os dois termos.

2.3.3 Caminho Computacional e o Tipo Identidade

Foi visto que o caminho computacional é uma entidade que torna natural o conceito de igualdade entre dois objetos computacionais. Um caminho computacional s entre dois objetos é a evidência que estabele a igualdade dos objetos. Também foi visto que essa é basicamente a ideia da igualdade proposicional e do tipo identidade. É um tipo cujos termos são a evidência que estabelece a igualdade. Portanto, dado essa semelhança, torna-se natural pensar que é possível formular completamente o tipo identidade a partir de caminhos computacionais. Naturalmente, a formulação será feita a partir de regras de inferência (QUEIROZ; OLIVEIRA, 2011):

$$\frac{A \text{ tipo} \qquad a : A \qquad b : A}{Id_A(a,b) \text{ tipo}} \, Id - F \qquad \frac{a =_s b : A}{s(a,b) : Id_A(a,b)} \, Id - I_1$$

A formação é exatamente igual a formulação anterior do tipo identidade. Entretanto, as diferenças já ficam evidentes a partir da própria regra de introdução. A única regra de introdução do tipo identidade usual é a reflexividade. Também foi visto que é exatamente essa limitação que causa todo o estranhamento e a complexidade desse tipo. Já na abordagem a partir de caminhos computacionais, os tipos identidades são introduzidos a partir de um caminho computacional. Ora, se existe um caminho que estabelece que $a =_s b$, então nada mais natural que $s(a,b)$ seja um elemento do tipo identidade. Além disso, a vantagem dessa introdução em relação à outra é clara: s tem a liberdade de ser um conjunto de aplicações dos axiomas discutidos anteriormente. Ou seja, além de encapsular a própria reflexividade, possui axiomas extremamente simples como transitividade e simetria, além de outros axiomas, também relativamente simples, que deixam os caminhos computacionais extremamente fáceis de serem manipulados. Isso ficará mais evidente na simplicidade da regra de eliminação. Antes, porém, é importante destacar que existe uma outra regra de introdução (QUEIROZ; OLIVEIRA, 2011):

$$\frac{a =_s b : A \qquad a =_t b : A \qquad s =_z t : Id_A(a,b)}{s(a,b) =_{\xi(z)} t(a,b) : Id_A(a,b)} \, Id - I_2$$

A regra acima mostra que dados dois caminhos s e t, se for possível estabelecer através de um caminho z que s e t são iguais, então o termo introduzido a partir de s será igual ao termo a partir de t, devido à regra ξ de igualdade definicional. Com essas regras de introdução já esclarecidas, é possível agora introduzir uma das regras mais importantes, a eliminação (QUEIROZ; OLIVEIRA, 2014a):

$$\frac{\begin{matrix} & [a =_g b : A] \\ m : Id_A(a,b) & h(g) : C \end{matrix}}{REWR(m, \acute{g}.h(g)) : C} \, Id - E_1$$

Antes de tudo, é necessário esclarecer que o $'$ no termo $\acute{g}$ indica que $\acute{g}$ é uma abstração, ou seja, a expressão espera uma entrada. Essa representação foi feita para não confundir essa abstração com a λ do λ-cálculo. Será visto que essa abstração possui propriedades semelhantes da abstração λ, mas, tecnicamente, não são a mesma coisa, já que a abstração λ possui equivalências definidas no λ-cálculo que podem não se aplicar.

Anteriormente, a regra de eliminação usava um recursor de difícil funcionamento conhecido como J. Apesar de ser possível chamar o recursor da regra vista acima também como J, o termo $REWR$ fica mais adequado, já que é um recursor com um funcionamento extremamente diferente. Ao contrário de J, $REWR$ funciona de forma extremamente simples. Se a partir de um caminho g que supõe-se existir e que estabelece que a é igual a b, é possível construir um termo $h(g)$ de um tipo C, então o recursor diz que a partir de qualquer termo m que também estabelece que a é igual a b é possível construir um elemento do tipo C. Ora, basta pegar m e substituir (ou aplicar) em $\acute{g}.h(g)$. Essa aplicação não está encapsulada na regra da eliminação. A regra se limita a construir um termo do tipo C, que é o $REWR(m,\acute{g}.h(g)) : C$. O funcionamento de $REWR$ ficará claro com as regras de redução. De qualquer forma, tente comparar essa eliminação com a do J. A grande desvantagem do J, além do funcionamento obscuro, é o fato que achar o ponto de partida é o maior problema. Como J é baseado apenas na reflexividade, achar o ponto de partida que justifica a construção do tipo torna-se uma tarefa não-trivial (esse ponto de partida também é comumente chamado de **o motivo** (HARPER, 2012)). Já para $REWR$, o ponto de partida, como pode-se ver claramente na dedução natural, é um caminho $a =_g b : A$. Por o ponto de partida ser um caminho, então é possível trabalhar com os axiomas vistos anteriormente, aplicando no caminho g. Dado a simplicidade dos axiomas, partir de g e chegar a $h(g) : C$ é geralmente uma tarefa simples. Após obter $h(g) : C$, o raciocínio está basicamente completo: basta aplicar diretamente a eliminação que será obtido o termo requerido. A naturalidade com que uma prova é feita a partir de $REWR$ será melhor explorada a partir de exemplos em uma subseção futura. Agora, a segunda forma de eliminação (QUEIROZ; OLIVEIRA, 2011):

$$\frac{\begin{array}{cc} & [a =_g b : A] \\ p =_r q : Id_A(a,b) & h(g) : C \end{array}}{REWR(p,\acute{g}.h(g)) =_{\mu(r)} REWR(q,\acute{g}.h(g)) : C}\, Id - E_2$$

A regra acima tem o papel de exibir uma das igualdades definicionais para os caminhos, e terá seu papel na obtenção de igualdades proposicionais entre caminhos. A regra acima apenas deixa claro que dado dois caminhos iguais, os $REWR$ construídos também serão iguais. É semelhante a ideia utilizada por $Id - I_2$. Com isso, existem agora as regras de redução e indução. Essas regras simulam a redução λ e a η respectivamente. Primeiro, será visto a redução (QUEIROZ; OLIVEIRA, 2011):

$$\frac{\dfrac{[a =_m b : A]}{m(a,b) : Id_A(a,b)}\, Id - I_1 \qquad \begin{array}{c} [a =_g b : A] \\ h(g) : C \end{array}}{REWR(m,\acute{g}.h(g)) : C}\, Id - E_1 \quad \rhd_\beta \quad \begin{array}{c} [a =_m b : A] \\ h(m/g) \end{array}$$

A regra acima é o que já foi comentado sobre a eliminação. Mostra como é possível computar a partir de *REWR*. Como foi dito, *REWR* funciona a partir de uma aplicação, ao substituir (aplicar) o caminho g por um caminho m em $g'.h(g)$, obtendo-se então a forma reduzida $h(m/g)$. Nota-se a semelhança dessa redução com a redução β do λ-cálculo. Por isso, essa redução pode ser considerada como equivalente a β da $\lambda\beta\eta$ igualdade. A seguir, a redução que simula o η (também chamada de *Id - Indução*) (QUEIROZ; OLIVEIRA, 2011):

$$\frac{e : Id_A(a,b) \qquad \dfrac{a =_t b : A}{t(a,b) : Id_A(a,b)}\, Id-I_1}{REWR(e,\acute{t}.t(a,b)) : Id_A(a,b)}\, Id-E_1 \quad \rhd_\eta \quad e : Id_A(a,b)$$

A regra acima, a *Id - Indução* é bastante simples. É usada caso o tipo construído seja o tipo mais trivial possível, isto é, a partir de um caminho t construir diretamente o termo $t(a,b) : Id_A(a,b)$. Ora, já que o próprio termo e é do tipo $Id(a,b)$, então pode-se ignorar o t e reduzir tudo a apenas $e : Id_A(a,b)$. Pela semelhança com a operação η da $\lambda\beta\eta$-igualdade, a redução recebe o nome de η, simulando, então, esta propriedade.

Com essas regras, a definição formal do tipo identidade a partir de um caminho computacional está concluída. Foi visto que a grande vantagem dessa abordagem é a flexibilidade e facilidade de uso da entidade principal, a qual é o caminho computacional. Aliado a isso, tem-se uma regra de introdução formulada com base na noção de **motivo**, junto com uma eliminação cujo operador, o chamado *REWR*, tem uma ação semelhante ao operador de eliminação do quantificador existencial (dado que a variável introduzida para denotar um caminho arbitrário, a saber, g, aparece em associação à suposiç ao introduzida pela regra), baseando-se nas próprias regras do λ-cálculo, como a β e η reduções. Dessa forma, ao construir um termo do tipo C, a transparência do processo em que o termo é construído passa a ser a grande vantagem dessa nova abordagem, em contrapartida à grande dificuldade e obscuridade de funcionamento do recursor J. Além disso, por ser baseado em uma teoria fundamentalmente computacional, como o λ-cálculo, o construtor *REWR* apresenta uma interpretação computacional natural. Na próxima seção, exemplos práticos de como construir tipos a partir das duas abordagens serão mostrados.

2.4 Reflexividade, Simetria e Transitividade do Tipo Identidade

Após as várias deduções naturais apresentadas nas seções anteriores, é natural surgir a dúvida sobre como utilizar as deduções naturais para construir termos de um determinado tipo. Em teoria dos tipos, dominar a forma com que termos são construídos é essencial. Isso se deve ao fato que, nessa teoria, provar a veracidade de A é exatamente a mesma coisa que construir um termo a tal que $a : A$. Mais do que isso, em teoria dos tipos, essa é a **única** forma de provar a veracidade de A. Olhando para o tipo identidade, logo fica claro que a ferramente mais poderosa

para construção de novos tipos são as regras de eliminação. Por isso foi dado tanto foco para J e $REWR$. Tendo isso em mente, o objetivo dessa subseção será, usando as duas abordagens, mostrar que é possível construir tipos que provam que o tipo identidade é reflexivo, simétrico e transitivo. Ao construir os tipos, uma comparação entre as duas abordagens será inevitável. O objetivo é comparar a complexidade e não o tamanho das deduções. Dado que a abordagem sem caminhos computacionais encapsula uma quantidade enorme de informações na regra de eliminação, as deduções tendem a ser menores que as que usam caminhos computacionais. Porém, esses é um dos motivos que acabam tornando a abordagem sem caminhos bem mais complexa.

Ao construir um novo tipo, é sempre necessário um ponto de partida. Como já foi dito anteriormente, esse ponto de partida é geralmente conhecido como 'o motivo' (HARPER, 2012). Como as operações que envolvem o ponto de partida são o único raciocínio necessário para construir um tipo (o restante é apenas aplicações diretas das regras de eliminação), então talvez a complexidade do ponto de partida e as operações deste sejam o melhor parâmetro que indica o quão complexo é cada abordagem. Os exemplos que serão provados são a reflexividade, a transitividade e a simetria. Essa escolha foi feita baseada no fato que são três propriedades fundamentais de qualquer teoria da igualdade, além de que, como será visto em capítulos posteriores, essas três propriedades simples são capazes de induzir estruturas matemáticas impressionantes.

2.4.1 Reflexividade

A reflexivdade é extremamente simples. Tão simples que já é naturalmente imbutida no tipo identidade sem caminhos computacionais. Ora, toda a teoria dessa abordagem é baseada na existência da reflexividade como entidade básica. Dessa forma, torna-se apenas necessário provar essa propriedade para a abordagem com caminhos computacionais. Formalmente, o objetivo é construir um termo para o tipo $\Pi_{(x:A)} Id_A(x,x)$. Ora, tambem não é necessário utilizar a regra de eliminação na abordagem com caminhos. O motivo é bastante simples, dado um termo $a:A$, basta aplicar o axioma da reflixividade para obter $a =_\rho a:A$:

$$\dfrac{\dfrac{\dfrac{[a:A]}{a =_\rho a:A}}{\rho(a,a):Id_A(a,a)}\, Id-I_1}{\lambda a.\rho(a,a):\Pi_{(a:A)} Id_A(a,a)}\, \Pi - I$$

Olhando a dedução acima, é perceptível que a única parte que exigiu alguma engenhosidade (nesse caso, um raciocínio extremamente simples) foi o **motivo**, que é $a =_\rho a$. O restante poderia, por exemplo, ser feito automaticamente por um computador.

2.4.2 Simetria

A simetria, apesar de não ser muito complicada, é um pouco mais complexa que a reflexividade. Primeiro, será visto como provar a simetria sem caminhos computacionais. Para ambas abordagens, o objetivo é o mesmo: construir um termo para o tipo $\Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))$. Como sempre, o primeiro passo é achar o motivo. Para a eliminação J, o motivo é muito mais complicado que simples caminhos computacionais. Olhando para a eliminação, logo percebe-se que a parte principal é achar um tipo $C(x,y,z)$ adequado, com $x:A$, $y:A$ e $z:Id_A(a,b)$. Ao contrário do processo usado para o tipo identidade com caminhos computacionais, achar o motivo não é tão simples como partir de um caminho e aplicar axiomas simples. De fato, até para o exemplo da simetria, achar o motivo para o caso sem caminhos computacionais pode demorar bastante tempo para alguém destreinado. O problema é que logo de início é necessário definir três variavéis, ao contrário de apenas uma usando caminhos computacionais. Além disso, não existe um processo fixo de como utilizar adequadamente tais variáveis. É necessário usar a intuição e, muitas vezes, tentativa e erro. Dessa forma, é possivel deduzir que o motivo é $C(x,y,z)$ como o tipo $Id_A(y,x)$, com $x:A$, $y:A$ e z como qualquer termo, já que é desnecessário para a construção de $Id_A(y,x)$ (Na dedução, z será representado por um traço $-$, indicando que é irrelevante). Com isso, é obtida a seguinte dedução natural:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a:A}\quad\dfrac{}{b:A}\quad\dfrac{}{c:Id_A(a,b)}\quad\dfrac{[x:A]}{r(x):Id(x,x)}\quad\dfrac{[x:A,y:A,-:Id_A(a,b)]}{Id_A(y,x)\ \text{tipo}}\,Id-E}{J(c,r(x)):Id(b,a)}}{\lambda c.J(c,r(x)):Id(a,b)\to Id(b,a)}}{\lambda b.\lambda c.J(c,r(x)):\Pi_{(b:A)}(Id_A(a,b)\to Id_A(b,a))}}{\lambda a.\lambda b.\lambda c.J(c,r(x)):\Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b)\to Id_A(b,a))}$$

Com isso, foi possível construir o termo a partir de J. Porém, poderia ser ainda mais dificil. A prova foi facilitada pelo fato que, ao escolher C como $Id_A(y,x)$, o termo $d(x):C(x,x,r(x))$ naturalmente se tornou $d:Id_A(x,x)$. Como $Id(x,x)$ é naturalmente habitado pela reflexividade, a prova tornou-se um pouco mais fácil. Agora, pode-se fazer a mesma prova usando caminhos computacionais. Ora, o motivo será muito mais simples. A partir de um caminho t, basta aplicar o axioma da simetria σ, obtendo-se o caminho inverso $\sigma(t)$. Obtém-se (QUEIROZ; OLIVEIRA, 2014a):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a=_t b:A]}{b=_{\sigma(t)}a:A}}{(\sigma(t))(b,a):Id_A(b,a)}\,Id-I_1\quad\dfrac{}{[p:Id_A(a,b)]}}{REWR(p,\acute{t}.\sigma(b,a)):Id_A(b,a)}\,Id-E_1}{\lambda p.REWR(p,\acute{t}.\sigma(b,a)):Id_A(a,b)\to Id_A(b,a)}}{\lambda b.\lambda p.REWR(p,\acute{t}.\sigma(b,a)):\Pi_{(b:A)}(Id_A(a,b)\to Id_A(b,a))}}{\lambda a.\lambda b.\lambda p.REWR(p,\acute{t}.\sigma(b,a)):\Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b)\to Id_A(b,a))}$$

Portanto, como pode-se ver, após descoberto o motivo, o restante fica direto e sem maiores dificuldades, apenas aplicando a eliminação e as abstrações necessárias.

2.4.3 Transitividade

Após as provas da simetria e da reflexividade, resta apenas provar a transitividade. Achar o motivo para a simetria não foi direto, mas também não foi algo complexo. A transitividade, porém, vai deixar claro a dificuldade em se achar o motivo sem usar caminhos computacionais. Por isso, será o exemplo que deixará mais claro as vantagens dos caminhos.

Como não poderia deixar de ser, o objetivo é construir um termo de um tipo que representa a transitividade a partir das regras de eliminação. No caso da transitividade, o tipo requerido é $\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))$. Primeiro, será provado a partir da abordagem sem caminhos. Ora, o primeiro passo é achar o motivo, ou seja, os valores de $x : A$, $y : A$ e $z : Id_A(x,y)$ e, a partir desses valores, propor um $C(x,y,z)$ adequado. Similar ao caso da simetria, não há uma forma padrão ou um conjunto de regras a se seguir, é necessário usar a intuição e tentativas e erro até chegar aos valores adequados. Esse processo torna-se um pouco complexo por causa da maior complexidade da transitividade, quando comparado à simetria e à reflexividade. Após esse processo, concluí-se que o um motivo adequado é: $x : A$, $y : A$, $- : Id_A(x,y)$ e $C(x,y,z) \equiv Id_A(y,c) \to Id_A(x,c)$. O termo z como $-$ significa que z vai ser irrelevante, pode ser qualquer coisa. Apesar de descoberto o motivo, não é possível ainda aplicar a eliminação. Isso por causa do termo $d(x)$. Ora, é necessário construir $d(x) : C(x,x,r(x)$ para poder usar a regra de eliminação. Nesse caso, $C(x,x,r(x)) \equiv Id_A(x,c) \to Id_A(x,c)$. Portanto, para construir $d(x)$, é necessário aplicar uma nova eliminação com um novo motivo. Porque não foi necessário essa segunda eliminação no caso da simetria? A resposta é que na simetria, tinha-se $C(x,x,r(x)) \equiv Id_A(x,x)$, o qual já é naturalmente habitado por $r(x)$. Processo semelhante acontecerá na construção de $d(x)$: o motivo será $x : A$, $y : A$, $- : Id_A(x,y)$ e $C'(x,y,z) \equiv Id_A(x,y)$. Nesse caso, $d'(x) : C'(x,x,r(x))$ e $C'(x,x,r(x)) \equiv Id_A(x,x)$, o qual é naturalmente habitado por $r(x)$. Segue a primeira eliminação que constrói $d(x)$:

$$\dfrac{\dfrac{\dfrac{x : A \quad c : A \quad q : Id_A(x,c) \quad r(x) : Id_A(x,x) \quad [x : A] \quad [x : A, y : A, - : Id_A(x,y)] \quad Id_A(x,y)\ \text{type}}{J(q,r(x)) : Id_A(x,c)}\, Id-E}{\lambda q.J(q,r(x)) : Id_A(x,c) \to Id_A(x,c)}$$

Com o termo $d(x)$ construído, é possível aplicar, finalmente, a eliminação que acarretará no termo desejado:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a : A \quad b : A \quad p : Id_A(a,b) \quad \lambda q.J(q,r(x)) : Id(x,c) \to Id(x,c) \quad [x : A] \quad [x : A, y : A, - : Id_A(x,y)] \quad Id_A(y,c) \to Id_A(x,c)\ \text{tipo}}{J(p,\lambda q.J(q,r(x))) : Id_A(b,c) \to Id_A(a,c)}\, Id-E}{\lambda p.J(p,\lambda q.J(q,r(x))) : Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c)}}{\lambda c.\lambda p.J(p,\lambda q.J(q,r(x))) : \Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}}{\lambda b.\lambda c.\lambda p.J(p,\lambda q.J(q,r(x))) : \Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}}{\lambda a.\lambda b.\lambda c.\lambda p.J(p,\lambda q.J(q,r(x))) : \Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}$$

Essa dedução natural finalmente termina a prova da transitividade para o tipo identidade sem caminhos computacionais. Como pode-se ver, foi necessário um grande esforço de raciocínio para descobrir os motivos que justificam a transitividade. Sendo a transitividade uma propriedade simples, esse exemplo mostra claramente os problemas da abordagem sem caminhos. Se até uma propriedade simples pode ter motivos complicados, imagine a dificuldade para provar propriedades mais complexas.

É possível, agora, pensar em como provar o mesmo fato para o tipo identidade com caminhos. Ora, o objetivo final é o mesmo: construir um termo para $\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))$. Naturalmente, é necessário achar o motivo. Usando caminhos computacionais, o motivo é extremamente simples e direto: dado um caminho t tal que $a =_t b$ e um outro u tal que $b =_u c$, é possível construir um caminho de a até c. Para comprovar isso, não é preciso nenhum argumento bem elaborado, basta usar diretamente os axiomas de igualdade da teoria dos tipos. Um dos axiomas, o axioma τ, já garante a transitividade. Portanto, basta fazer $a =_{\tau(t,u)} c$. Esse é todo o motivo necessário para a eliminação. O restante sao apenas passos diretos (QUEIROZ; OLIVEIRA, 2011):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a =_t b] \quad [b =_u c]}{a =_{\tau(t,u)} c}}{(\tau(t,u))(a,c) : Id_A(a,c)}\,Id-I_1 \quad s(b,c) : Id_A(b,c)}{REWR(s(b,c),\acute{u}(\tau(t,u))(a,c)) : Id_A(a,c)}\,Id-E_1 \quad [w(a,b) : Id_A(a,b)]}{REWR(w(a,b),\acute{t}REWR(s(b,c),\acute{u}(\tau(t,u))(a,c))) : Id_A(a,c)}\,Id-E_1}{\lambda s.REWR(w(a,b),t'REWR(s(b,c),\acute{u}(\tau(t,u))(a,c))) : Id_A(b,c) \to Id_A(a,c)}}{\lambda w.\lambda s.REWR(w(a,b),\acute{t}REWR(s(b,c),\acute{u}'(\tau(t,u))(a,c))) : Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c)}}{\lambda c.\lambda w.\lambda s.REWR(w(a,b),t'REWR(s(b,c),\acute{u}(\tau(t,u))(a,c))) : \Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}}{\lambda b.\lambda c.\lambda w.\lambda s.REWR(w(a,b),\acute{t}REWR(s(b,c),\acute{u}(\tau(t,u))(a,c))) : \Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}}{\lambda a.\lambda b.\lambda c.\lambda w.\lambda s.REWR(w(a,b),\acute{t}REWR(s(b,c),\acute{u}(\tau(t,u))(a,c))) : \Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}$$

A dedução natural acima constrói o termo necessário. Apesar da dedução ter ficado extensa, cada passo é automático, sem precisar de grande esforço em termos de raciocínio. Todos os passos derivam diretamente do motivo, que, por sua vez, deriva de um axioma simples da igualdade da teoria dos tipos. Como o tipo a ser construído é extenso, então é natural que a dedução natural também seja extensa.

O tamanho das deduções remete ao ponto discutido na introdução dessa seção. Como o tamanho da dedução é geralmente conseqüência de aplicação de diversas regras diretas, então o melhor parâmetro para comparar a dificuldade das duas abordagens é a dificuldade de se obter o motivo. Os motivos das deduções envolvendo o tipo identidade sem caminhos mostraram-se complicados de serem obtidos, já que não dependia do uso de regras já fixadas. Além disso, em um exemplo um pouco mais complexo, como o da transitividade, foi necessário aplicar a eliminação duas vezes e obter dois motivos diferentes. Comparado a isso, os motivos da abordagem com caminhos são extremamente mais simples, por derivarem de axiomas intuitivos, como os da reflexividade, simetria e transitividade.

2.5 Considerações Finais Sobre Teoria dos Tipos

Por o tipo identidade ser um conceito fundamental da teoria dos tipos, mostrar os principais conceitos dessa teoria foi essencial. Além disso, conceitos como o operador Π foram extensivamente usados nas deduções naturais. Com os conceitos básicos bem estabelecidos, o tipo identidade foi formalmente definido. Foram vistas as abordagens intensional e extensional. As dificuldades da abordagem atual do tipo identidade, a qual passou a ser chamada de abordagem sem caminhos computacionais, foram apontadas. Como dificuldade principal, foi apontado a não-intuitividade e o difícil funcionamento do operador J. Diante desse contexto, foi mostrada uma abordagem que define o tipo identidade a partir de uma entidade conhecida como caminhos computacionais. Foram mostradas as vantagens em termos de simplicidade dessa nova abordagem. Ao utilizar o tipo identidade na prática, provando-se a reflexividade, simetria e transitividade, a vantagem da abordagem com caminhos ficou ainda mais evidente.

Com esse capítulo, o primeiro objetivo do trabalho foi completamente concluído. Foi possível definir formalmente o tipo identidade a partir de caminhos e mostrar claramente as vantagens. Com isso, resta alcançar o segundo objetivo, que é encontrar e desenvolver propriedades matemáticas para o conceito de caminhos computacionais. Como foi visto nesse capítulo, caminhos computacionais foram formalmente definidos a partir do ponto de vista computacional. Entretanto, o significado matemático de um caminho computacional não foi obtido. Como a teoria dos tipos tem uma vertente fortemente matemática, alcançar uma interpretação matemática é essencial. Esse será o foco dos próximos capítulos.

3

Teoria das Categorias

A partir desse capítulo, a parte matemática dos caminhos computacionais começará a ser desenvolvida. A teoria que será utilizada para construir essa modelagem matemática é chamada de teoria das categorias. Será visto que a teoria das categorias é extremamente poderosa, mas possuí um alto grau de complexidade. Por isso, esse capítulo se limitará a tentar expor as peculiaridades dessa teoria da forma mais simples possível. Para obter esse objetivo, serão apresentados diversos conceitos básicos que são fundamentais. A partir dos conceitos básicos, serão apresentados conceitos mais avançados, os quais serão de grande importância para entender a interpretação do conceito de caminhos computacionais usando categorias. Essa interpretação, por sua vez, não aparecerá nesse capítulo. Isso se deve ao fato que a interpretação possui diversas pecularieddades. Como os conceitos da teoria das categorias já são complexos, é mais sensato deixar a interpretação para capítulos posteriores.

A teoria das categorias apareceu pela primeira vez a partir dos trabalhos dos matemáticos Eilenberg & Mac Lane em 1945. Inicialmente, a teoria das categorias surgiu como uma ferramenta para estudar transformações naturais e funtores (MARQUIS, 2014). Os funtores e as transformações naturais são basicamente entidades que transformam uma estrutura matemática em outra. Por exemplo, transformar uma função entre conjuntos em uma homotopia de espaços topológicos. Apesar de ter surgido como uma ferramenta para estudar essas transformações, a teoria das categorias rapidamente tomou proporções muito maiores. Hoje é amplamente usada tanto na matemática como na computação. Ora, o próprio conceito de caminho computacional, o qual é intrisicamente um conceito da computação, vai ser modelado matematicamente pela teoria das categorias. Até a própria lógica e o λ-cálculo podem ser modelados usando categorias. Além disso, a teoria das categorias ganha cada vez mais força como fundamento da matemática, substituindo o ZFC (MARQUIS, 2014).

O poderio da teoria das categorias, porém, não substitui a importância da teoria dos tipos. Apesar de poder ser utilizada como base da matemática, as categorias compartilham alguns problemas do ZFC, como o fato de não ser uma teoria facilmente modelável através de computadores. A importância das categorias para esse trabalho é o fato que essa teoria possui fortes ligações com a teoria dos tipos. Isso fica evidente devido ao fato que diversas estruturas da

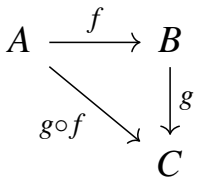
teoria dos tipos podem ser modeladas matematicamente através de categorias.

Por fim, vale ressaltar que esse capítulo se limitará a estudar apenas a teoria das cateogorias tradicional. A necessidade de afirmar isso é devido ao fato da existência de uma teoria chamada de teoria das categorias de alta ordem. Será visto no capítulo 5 que essas categorias de alta ordem tem uma grande importância para esse trabalho. A teoria das categorias já tem um alto grau de dificuldade, e, comparativamente, a teoria das categorias de alta ordem é ainda muito mais complexo. Portanto, por motivos de sensatez, os conceitos necessários dessa teoria mais avançada foram alocados para o capítulo 5.

3.1 Conceitos básicos

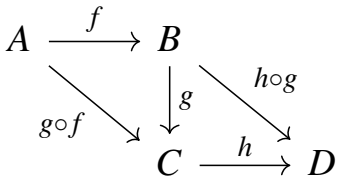
A teoria das categorias tem uma peculiaridade interessante quando comparada à teoria dos tipos ou ZFC. Tanto a teoria dos tipos quanto o ZFC, o conceito básico das respectivas teorias são conceitos fundamentais. Não faz sentido perguntar o que é um tipo ou um conjunto, mas sim o que pode ser construído a partir dessas entidades. Já no caso das categorias, é exatamente o contrário. É possível provar, a partir de regras e entidades bem estabelecidas, se uma estrutura é ou não é uma categoria. Porém, antes de formalmente introduzir o que é uma categoria, talvez a forma mais didática seja analisar as funções. O motivo disso é que pode-se dizer que uma categoria é construída a partir de estruturas extremamente similares às funções, as quais são chamadas de morfismos.

Sejam A, B e C conjuntos e f e g funções tal que $f : A \to B$ e $g : B \to C$. É possível, então, formar uma função composta $(g \circ f) : A \to C$, definida por $(g \circ f) = g(f(x))$. Através de um diagrama, é possível mostrar essas funções:



O diagrama acima foi mostrado devido ao fato que é extremamente comum trabalhar com diagramas desse tipo em teoria das categorias. Categorias são baseadas em objetos (no caso acima, os conjuntos) e setas ou morfismos (no caso acima, as funções). Dessa forma, é natural que essa estrutura seja representada por diagramas. Os diagramas não são meramente ilustrações de uma estrutura. São frequentemente usados para provar propriedades importantes. Muitas vezes, o próprio diagrama servirá como argumento.

Continuando com as funções, uma propriedade interessante pode ser notada: a composição é associativa. Em outras palavras, se existe um $h : C \to D$, então $(h \circ g) \circ f = h \circ (g \circ f)$. A propriedade é melhor representa pelo seguinte diagrama (AWODEY, 2010):



Por fim, funções ainda tem mais uma propriedade interessante. Para cada conjunto A, existe uma função identidade $1_A : A \to A$ definida por $1_A(a) = a$. A função identidade funciona como o elemento neutro da composição. Em outras palavras, tem-se que $1_B \circ f = f = f \circ 1_A$. Na próxima subseção será dado uma definição formal do conceito de categoria, o qual será uma generalização para essas propriedades de funções.

3.1.1 Categorias

O conceito fundamental da teoria das categorias. Segue a definição:

Definição 3.1 (Categorias (AWODEY, 2010))**.** *Uma categoria* $\mathbf{C}$ *é uma estrutura com os seguintes elementos e regras:*

- *Objetos: Os objetos são geralmente representados por letras maíusculas,* $A, B, C....$ *O conjunto de todos os objetos é chamado de* C_0.
- *Setas(morfismos): As setas ou morfismos são comumente representadas por letras que são usadas para representar funções, como* $f, g, h....$ *É importante ressaltar que as setas são entre dois objetos. O conjunto de todas as setas é chamado de* C_1.
- *Domínio e codomínio: Dado uma seta* $f : A \to B$*, o domínio de* f *é* A *e é denotado por* $dom(f) = A$*. O codomínio de* f *é* B *e é denotado por* $cod(f) = B$.
- *Composição: Dado setas* $f : A \to B$ *e* $g : B \to C$*, sempre existe uma seta* $h = (g \circ f) : A \to C$*. Essa seta é chamada de composição de* f *com* g*. Além disso,* h *é universal, ou seja, dado um* $m = (g \circ f)$*, então* $m = h$.
- *Seta identidade: Para todo objeto* A *existe uma seta* $1_A : A \to A$ *chamada de seta identidade.*
- *Regra da associatividade: Dado setas* $f : A \to B$*,* $g : B \to C$ *e* $h : C \to D$*, então sempre é verdade que* $(h \circ g) \circ f = h \circ (g \circ f)$.
- *Regra da identidade: Dado uma seta* $f : A \to B$*, tem-se que* $1_B \circ f = f = f \circ 1_A$.

A partir da definição acima, fica claro que funções seguem uma estrutura de categoria. De fato, uma das categorias mais clássicas é uma geralmente conhecida por **Sets**. Nessa categoria, os objetos são os conjuntos e as setas são as funções entre os conjuntos. Como visto anteriormente, as funções obedencem as regras de associatividade e de identidade e, portanto, **Sets** é claramente uma categoria. Porém, antes de continuar com outros conceitos, talvez seja interessante mostrar

mais um exemplo para deixar claro como usar a definição acima para provar que uma estrutura é uma categoria. Um exemplo interessante é a estrutura **Pos**. Como foi visto no *capítulo 1*, é possível ordenar um conjunto a partir de uma relação de ordem $\leq$. Esses conjuntos ordenados são chamados de *posets* (conjuntos parcialmente ordenados). Dessa forma, nessa estrutura um objeto é um poset e uma seta é uma função monótona (AWODEY, 2010). É dito que $f : A \to B$ é monótona se $a \leq b \Rightarrow f(a) \leq f(b)$.

Proposição 3.1. ***Pos*** *é uma categoria.*

Demonstração: Uma prova que uma estrutura é uma categoria geralmente segue um conjuntos de passos bem definidos. Inicialmente, é necessário definir o que é uma seta, as composições e a seta identidade. Após isso, basta checar as regras de associatividade e identidade:

- Composição: A composição é dada da mesma forma que em **Sets**. Ou seja, dados $f : A \to B$ e $g : B \to C$, então pode-se definir $(g \circ f) = g(f(x))$. Ora, pra essa composição fazer sentido, é necessário que $(g \circ f)$ seja monótona. Suponha $a \leq b$. Como f é monótona, então $f(a) \leq f(b)$. Por sua vez, como g é monótona, então $g(f(a)) \leq g(f(b))$ e, portanto, $g \circ f$ é monótona.
- Identidade: A seta identidade também será a mesma que no caso de **Sets**. Ou seja, será uma $1_A : A \to A$ definida por $1_A(a) = a$. Ora, 1_A é clarametne monótona, pois se $a \leq b$, então $1_A(a) = a \leq b = 1_A(b)$.
- Associatividade: Já que setas são funções, a associatividade já é garantida.
- Regra de Identidade: Mesma coisa que no caso da associatividade. Já foi visto, para o caso de funções, que 1_A segue as regras de identidade.

□

Portanto, **Pos** é uma categoria. A escolha por mostrar **Pos** foi proposital, já que é uma categoria de extrema importância para a computação. O motivo disso é que **Pos** é extremamente utilizada ao formalizar a álgebra booleana. Essa formalização, porém, não é um dos escopos desse trabalho. Com o conceito de categoria bem estabelecido, já é possível aprender novos conceitos interessantes.

3.1.2 Isomorfismo

Nas diversas áreas da matemática, é muito comum trabalhar com duas estruturas que têm propriedades semelhantes, mas que são diferentes. Com a necessidade de estabelecer uma ligação entre essas duas estruturas, foi introduzido o conceito de isomorfismo. O problema é que a definição formal de isomorfismo é intriscamente dependende da estrutura que está sendo estudada. Por exemplo, o conceito de isomorfismo de conjuntos é diferente do de posets, o qual é

diferente do isomorfismo de grupos, sendo todos também diferentes do conceito de isomorfismo para espaços topológicos. Diante disso, a possibilidade de unificar essas definições distintas para o mesmo conceito seria uma grande vantagem. Isso torna-se possível através do uso da teoria das categorias. Talvez essa unificação do conceito de isomorfismo seja uma das melhores formas de demonstrar o poderio dessa teoria. Segue a definição:

Definição 3.2 (Isomorfismo (AWODEY, 2010))**.** *: Em uma categoria* $\mathbf{C}$*, uma seta* $f : A \to B$ *é chamada de isomorfismo se existe uma seta* $g : B \to C$ *tal que* $g \circ f = 1_A$ *e* $f \circ g = 1_B$*. Nessa ocasião,* g *é chamada de inversa de* f*. É dito que* A *é isomorfo a* B *e a notação é dada por* $A \cong B$*. A seta que gerou o isomorfismo é geralmente denotada por:* $f : A \xrightarrow{\sim} B$*.*

Proposição 3.2. *Seja a seta* f *um isomorfismo e* g *a inversa de* f*. Então,* g *é única.*

Demonstração: Seja h uma inversa de f. Para provar que g é única, é suficiente provar que $h = g$. As seguintes equações estabelecem esse resultado:

$$\begin{aligned}(g \circ f) \circ h &= g \circ (f \circ h)\\ 1_A \circ h &= g \circ 1_B\\ h &= g.\end{aligned}$$

A primeira equação é claramente justificada pela associatividade da composição. A segunda pelo fato que g e h são inversas de f e a última em razão do fato de que a composição segue as regras da identidade. □

Devido à prova acima, é possível criar uma notação para indicar a inversa de um isomorfismo f. Como a inversa g é única, pode-se denotar g como f^{-1}.

Olhando para a definição de isomorfismo, o que seria um isomorfismo em **Sets**?. Ora, para que os conjuntos A e B sejam isomorfos, é necessário uma função $f : A \to B$ e outra $g : B \to A$ tal que $g(f(x)) = x$. Pela proposição acima, $g = f^{-1}$. Ou seja, g é a inversa de f. Ora, sabe-se que uma função f possui inversa se e somente se f é bijetiva. Portanto, a condição necessária e suficiente para que A e B sejam conjuntos isomorfos em **Sets** é que exista uma função bijetiva f entre os dois conjuntos. Essa interpretação originada da teoria das categorias produz resultados interessantes:

Proposição 3.3. *A e B são isomorfos em **Sets** se e somente se possuem a mesma cardinalidade (informalmente, quer dizer que os conjuntos tem o mesmo tamanho).*

Demonstração: Tanto a prova da ida como a da volta são diretas. Ora, sabemos que se A e B são isomorfos, então existe uma bijeção $f : A \to B$. Como, por definição, dois conjuntos tem a mesma cardinalidade se existe uma função bijetiva entre eles, então, por f, A e B tem o mesmo tamanho. Se os dois conjuntos tem o mesmo tamanho então, também pela definição de cardinalidade, existe uma f bijetora entre os conjuntos. Logo, como essa é a condição necessária e suficiente para que A e B sejam isomorfos, eles então o são. □

É um resultado bastante conhecido da teoria dos conjuntos que $\mathbb{N}$, $\mathbb{Z}$ e $\mathbb{Q}$ tem a mesma cardinalidade (HRBACEK; JECH, 1999). Pela proposição acima, então, os naturais são isomorfos aos inteiros e também aos racionais. Isso pode causar grande estranhamento, pois como é possível os naturais terem a mesma estrutura dos inteiros? Ora, lembre-se que em **Sets** o objeto são apenas conjuntos e não os conjuntos com uma estrutura de ordem. É óbvio que, se considerada as estruturas de ordem tradicionais, o conjunto ordenado dos naturais não é isomorfo aos dos inteiros. Ora, só o fato de os naturais terem menor elemento e os inteiros não, já quebra qualquer possível isomorfismo. Considerando, claro, as ordenações tradicionais.

O que seria, então, um isomorfismo em **Pos**? Ora, já que uma seta é uma função monótona, é natural pensar que um isomorfismo é uma função bijetora monótona. Isso é fácil de checar. Se f é uma função bijetora monótona, basta checar que f^{-1} também será injetora monótona. Ora, se $a \leq b$, então $f(a) \leq f(b)$ e, portanto $a = f^{-1}(f(a)) \leq f^{-1}(f(b)) = b$. Logo, f^{-1} é monótona.

A partir dessa definição, é possível mostrar que, dependendo da ordenação, os naturais podem ser isomorfos aos inteiros:

Demonstração: Considere os naturais junto com a ordenação tradicional, ou seja, $(\mathbb{N}, \leq) = 0, 1, 2, \ldots$. Para os inteiros, considere uma ordenação $<''$ tal que $(\mathbb{Z}, <'') = 0, -1, 1, -2, 2, -3, 3 \ldots$. É necessário achar uma f bijetiva tal que $f(0) = 0$, $f(1) = -1$, $f(2) = 2$, $f(3) = -2$, etc. Basta fazer $f(x) = (-1)^x . \lceil \frac{x}{2} \rceil$. f é claramente monótona, ou seja, $a \leq b \Rightarrow f(a) <'' f(b)$, além de ser bijetiva. Logo, por f, $(\mathbb{N}, \leq) \cong (\mathbb{Z}, <'')$. □

Com isso, foi visto como a teoria das categorias generaliza o importante conceito de isomorfismo entre objetos matemáticos. Além disso, foi mostrado exemplos práticos de isomorfismo para **Sets** e **Pos**. Será visto que isomorfismos serão um dos conceitos essenciais para a interpretação dos caminhos computacionais.

3.1.3 Grupóide

Grupóide é uma estrutura baseada no conceito de isomorfismo, dada pela seguinte definição:

Definição 3.3 (Grupóide (AWODEY, 2010))**.** *Seja* **C** *uma categoria. É dito que* **C** *é uma grupóide caso toda seta $f \in C_1$ é um isomorfismo. Ou seja, todos os objetos de* **C** *são isomorfos entre si.*

O conceito de grupóide tem se revelado de grande utilidade no desenvolvimento da semântica do tipo identidade. Se uma categoria **C** é uma grupóide, então todos os objetos de **C** tem estruturas semelhantes entre si. A bem da verdade, o conceito de grupóide ganhou uma relevância enorme com a descoberta que, na teoria de tipos de Martin-Löf, os tipos seguem, de certa forma, uma estrutura de grupóide (HOFMANN; STREICHER, 1994). De fato, é possível provar que os tipos tem uma interpretação categórica que condiz com o conceito de grupóide em

um sentido fraco, pois as regras de grupóide são satisfeitas usando um tipo especial de igualdade. Os detalhes sobre essa interpretação e o sentido exato do termo 'fraco' serão vistos no capítulo 4. Além disso, o grande objetivo do capítulo 4 será mostrar que o conceito de caminhos de reescritas também podem ser interpretados através de grupóides. E mais ainda, também seguirão uma estrutura 'fraca'. Ou seja, a interpretação dos caminhos computacionais será condizente com a interpretação proposta por HOFMANN; STREICHER (1994), que surgiu como interpretação da formulação original do tipo identidade intensional dado por Martin-Löf. Por isso, o conceito de grupóide talvez seja o conceito mais importante do presente trabalho.

Deixando os conceitos mais complicados para o capítulo 4, é possível dar um exemplo simples de grupóide utilizando uma categoria simples. Basta tomar uma categoria em que todos os objetos são conjuntos do mesmo tamanho. Dessa forma, como já foi provado, todos serão isomorfos entre si. Um bom exemplo em **Pos** é um pouco mais difícil, já que envolve alguns conceitos mais elaborados de teoria dos conjuntos. Porém, para um leitor mais familiarizado com essa teoria, basta tomar uma categoria formada por objetos $(A_n, \leq_n)$ em que todos os conjuntos A_n têm o mesmo tamanho e que os $\leq_n$ são boa-ordenações isomorfas ao mesmo número ordinal. Por exemplo, no caso provado para $(\mathbb{N}, \leq)$ e $(\mathbb{Z}, <'')$, ambas as boa-ordenações eram isomorfas ao ordinal ω. Dessa forma, todos os $(A_n, \leq_n)$ serão isomorfos entre si.

3.1.4 Funtores

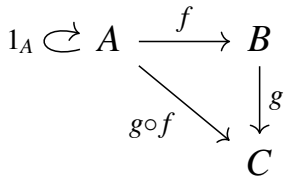
A noção de funtor é um dos conceitos mais importantes em teoria das categorias. Os funtores podem ser entedidos como uma função em que o objeto é uma categoria. É uma forma de transformar uma categoria em uma outra. Por exemplo, pense na transformação de uma categoria **C** em uma categoria **D**. Para transformar **C** em **D**, é necessário transformar os objetos de **C** nos objetos de **D** e as setas de **C** nas setas de **D**. Além disso, as propriedades de associatividade e identidade devem ser preservadas. Segue a definição formal:

Definição 3.4 (Funtor (AWODEY, 2010))**.** *Um funtor $F : \mathbf{C} \to \mathbf{D}$ entre as categorias **C** e **D** é um mapeamento entre objetos e setas tal que:*

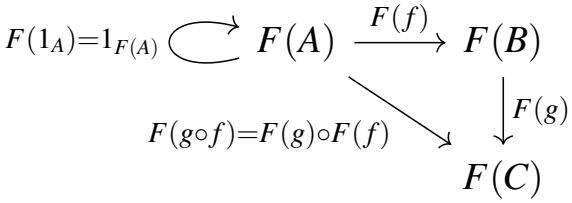
- $A \in C_0$ *é mapeado para* $F(A) \in D_0$,
- $f : A \to B \in C_1$ *é mapeado para* $F(f) : F(A) \to F(B) \in D_1$,
- $F(g \circ f) = F(g) \circ F(f)$,
- $F(1_A) = 1_{F(A)}$.

Talvez a definição acima seja mais fácil de ser visualizada através de diagramas. Portanto, segue o exemplo (AWODEY, 2012):

Exemplo 3.1. O exemplo mostrará a representação gráfica da ação de um funtor genérico $F : \mathbf{C} \to \mathbf{D}$. Seja **C** uma categoria representada pelo seguinte diagrama:



Ao aplicar F, será obtida uma $F(\mathbf{C}) = \mathbf{D}$ representada pelo seguinte diagrama:



O diagrama acima é auto-explicativo, é basicamente uma representação gráfica da definição de funtor.

É muito comum encontrar funtores na prática. Um exemplo simples é o funtor For : **Pos** $\rightarrow$ **Sets**. For pode ser definido por sua ação nos objetos e setas de **Pos**. Como é sabido, um objeto de **Pos** é um par $(A, \leq)$, sendo A um conjunto. For age da seguinte forma: $For(A, \leq) = A$. Nas setas, tem-se que $For(f : (A, \leq) \rightarrow (B, \leq')) = f : A \rightarrow B$. Basicamente, For é um funtor que 'esquece' a ordenação de um conjunto ordenado, retornando apenas um conjunto. Para as setas, For esquece que f é uma função monótona, retornando apenas uma função. Esse tipo de funtor que elimina (ou 'esquece') informações de uma estrutura mais complexa, transformando em uma estrutura mais simples são muito comuns na matemática. São conhecidos como funtores esquecedores. Um outro exemplo bastante conhecido é For : **Grp** $\rightarrow$ **Sets**, o qual recebe um grupo e esquece a estrutura de grupo, retornando apenas o conjunto base do grupo. Apesar da estrutura simples, o funtor esquecedor tem grande importância teórica, já que foi a base para uma entidade extremamente interessante conhecida com adjuntor. Infelizmente, o estudo de adjuntores está fora do escopo desse trabalho.

Com isso, é finalizado a parte de funtores. Os funtores serão especialmente importantes para definir formalmente uma 2-categoria, como será visto no capítulo 5.

3.1.5 Dualidade

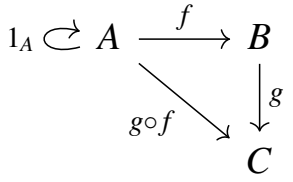
A dualidade é um princípio de teoria das categorias que afirma que, para qualquer categoria **C**, existe uma categoria oposta chamada de C^{OP}. A categoria C^{OP} é definida da seguinte forma (AWODEY, 2012)

Definição 3.5. *Dado uma categoria* $\mathbf{C}$*, a categoria* $\mathbf{C}^{OP}$ *é a categoria oposta a* $\mathbf{C}$ *definida por:*

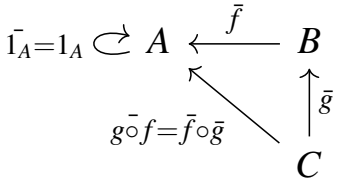
- $\boldsymbol{C}_0^{OP} = \boldsymbol{C}_0$.
- $\boldsymbol{C}_1^{OP} = \boldsymbol{C}_1$.

- $dom\ \mathbf{C}^{OP} = cod\ \mathbf{C}$.
- $cod\ \mathbf{C}^{OP} = dom\ \mathbf{C}$.

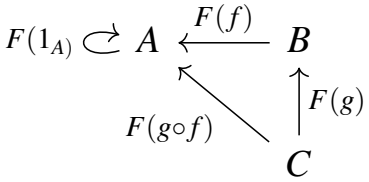
A dualidade pode ser representada graficamente. Seja a categoria **C** representada pelo seguinte diagrama:



Então, C^{OP} será dada por:



Outra parte importante da categoria oposta é que é possível definir, a partir dela, um novo tipo de funtor chamado de funtor contravariante. O functor contravariante é dado por $F : \mathbf{C}^{OP} \to \mathbf{D}$. Um funtor contravariante age de forma similar a um funtor normal. Mesmo o funtor agindo em $\mathbf{C}^{OP}$, é comum aplicar o funtor contravariante na própria categoria **C**. Para isso, basta inverter a ordem nas setas após a aplicação do funtor. Considerando **C** como a mesma categoria definida acima, o diagrama da aplicação do funtor contravariante F fica o seguinte:

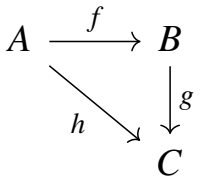


É bastante comum o uso de funtores contravariantes em teoria das categorias. O grande poder da dualidade é o fato que diversas propriedades são obtidas a partir da categoria oposta de uma outra propriedade. Por exemplo, será visto que a soma entre dois objetos de uma categoria é exatamente o produto na categoria oposta.

3.1.6 Comutatividade

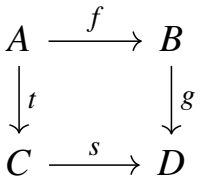
Essa breve subseção é dedicada a duas construções extremamente usadas em teoria das categorias: os diagramas de triângulos e quadrados comutativos.

Definição 3.6 (Triângulo comutativo)**.** *O triângulo comutativo é o seguinte diagrama:*



Para ser comutativo, o triângulo ABC deve obedecer a seguinte equação: $g \circ f = h$.

Definição 3.7 (Quadrado comutativo)**.** *O quadrado comutativo é o seguinte diagrama:*



Para ser comutativo o quadrado ABCD deve, então, obedecer a seguinte equação: $g \circ f = s \circ t$.

Diversas entidades e provas da teoria das categorias dependem de triângulos e quadrados comutativos. Por isso, são de grande importância.

3.1.7 Produto Entre Duas Categorias

Antes de definir o produto entre categorias, é importante ressaltar que existe uma diferença entre produto entre categorias e produto entre objetos de uma categoria. O produto entre categorias é uma operação que constrói uma nova categoria, enquanto o produto entre objetos de categoria produz um novo objeto a partir do produto entre outros dois objetos. Além disso, sempre é possível obter um produto entre categorias. Já no caso de produto entre objetos, algumas vezes o produto não existe. Essa subseção será focada no produto entre categorias. O produto entre objetos merece uma atenção especial, já que tem uma forte ligação com o tipo produto da teoria dos tipos. Portanto, será detalhado posteriormente em uma nova seção.

Definição 3.8 (Produto entre duas categorias (AWODEY, 2010))**.** *Dado as categorias* $\mathbf{C}$ *e* $\mathbf{D}$*, é possível obter uma categoria* $\mathbf{C} \times \mathbf{D}$ *definida por:*

- *Objetos: objetos são da forma* (C,D)*, com* $C \in C_0$ *e* $D \in D_0$.
- *Setas: setas são da forma* $(f,g) : (C,D) \to (C',D')$*, com* $f : C \to C' \in C_1$ *e* $g : D \to D' \in D_1$.
- *Composição: A composição é definida por componentes, sendo dada por* $(f',g') \circ (f,g) = (f' \circ f, g' \circ g)$.
- *Identidade: A seta identidade também é definida componente-a-componente, dada por* $1_{(C,D)} = (1_C, 1_D)$.
- *Dois funtores-projeção, representadas pelo seguinte diagrama:*

$$\boldsymbol{C} \xleftarrow{\pi_1} \boldsymbol{C}\times\boldsymbol{D} \xrightarrow{\pi_2} \boldsymbol{D}$$

As projeções são definidas por $\pi_1(C,D) = C$, $\pi_2(C,D) = D$, $\pi_1(f,g) = f$ *e* $\pi_2(f,g) = g$.

A definição é bastante clara e será essencial para definir uma bicategoria no capítulo 5.

3.1.8 Hom e Categorias Pequenas

Em teoria das categorias, existe uma estrura de grande importancia chamada de *Hom*:

Definição 3.9 ($Hom_{\mathbf{C}}(A,B)$ (AWODEY, 2012))**.** *$Hom_{\boldsymbol{C}}(A,B)$ é a estrutura que contém todas as setas partindo do objeto A e B, ambos objetos de* $\boldsymbol{C}$.

O termo utilizado foi o de estrutura, já que é possível que a estrutura seja algo maior que um conjunto. Em categorias de alta ordem, por exemplo, a estrutura é uma categoria. Também referindo-se a tamanho de uma categoria, é possível definir categorias grandes e pequenas:

Definição 3.10 (Categoria grande e pequena (AWODEY, 2010))**.** *Uma categoria* $\boldsymbol{C}$ *é chamada de pequena se tanto* C_0 *e* C_1 *são conjuntos. Caso contrário, a categoria é chamada de grande.*

O conceito de categoria grande surgiu do fato que muitas categorias são demasiadamente grandes para os objetos e setas serem representados por conjuntos. Algumas vezes, os próprios objetos de uma categoria formam outra categoria. O mesmo acontece com as setas. Além disso, a necessidade de distinguir uma categoria grande de uma pequena é que diversas propriedades de categorias só funcionam para categorias pequenas. Isso acontece porque essas propriedades geralmente dependem diretamente do fato que os objetos e setas são conjuntos. Além de categoria grande e pequena, existe mais uma definição:

Definição 3.11 (Categoria localmente pequena (AWODEY, 2010))**.** *Uma categoria* $\boldsymbol{C}$ *é dita localmente pequena se, para qualquer par de objetos* $A,B \in C_0$*, tem-se que* $Hom_{\boldsymbol{C}}(A,B)$ *é um conjunto.*

A distinção acima foi criada em razão do fato de que, algumas vezes, não é necessário que uma categoria **C** seja pequena, mas que apenas seja localmente pequena. Exemplos de categorias que não são localmente pequenas ficará mais claro no capítulo 5, com o estudo de algumas categorias de alta ordem.

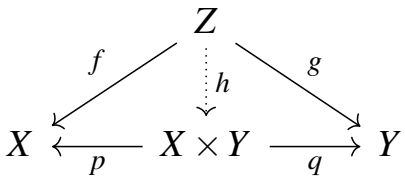
3.2 Produto em uma categoria

Na seção anterior foi visto a definição de produto entre categorias. Essa seção tem o objetivo de definir o produto entre objetos dentro de uma categoria. A importância dessa seção está no fato que ela mostrará claramente a conexão direta entre teoria dos tipos e a

teoria das categorias. É exatamente essa conexão estreita entre as duas teorias que motivou a tentativa explorar as propriedades matemáticos de caminhos computacionais a partir da teoria das categorias.

Definição 3.12 (Produto em uma categoria (NLAB, 2014))**.** *Em uma categoria* $\mathbf{C}$*, o produto entre dois objetos* $X, Y \in C_0$ *é um objeto* $X \times Y$ *equipado com morfismos* $p : X \times Y \to X$ *e* $q : X \times Y \to Y$*, os quais são chamados de projeções. Além disso, o produto possui uma propriedade universal, a qual diz que dado um objeto* $Z \in C_0$ *e mapas* $f : Z \to X$ *e* $g \to Z \to Y$*, então existe um mapa único* $h : Z \to X \times Y$ *tal que* $p \circ h = f$ *e* $q \circ h = g$*.*

Graficamente, o produto é representado da seguinte forma (AWODEY, 2010):



Os dois triângulos acima são comutativos. Olhando a definição, a parte das projeções é intuitiva de entender. Porém, a parte que fala sobre Z e a unicidade de h talvez seja um pouco menos intuitiva. Porém, a explicação é simples. Se existe uma seta $f : Z \to X$ e $g : Z \to Y$, então a unicidade de $h : Z \to X \times Y$ apenas indica que só existe uma única forma de construir o produto através dessas duas setas. Dessa forma, pode-se denotar que $h = (f, g)$.

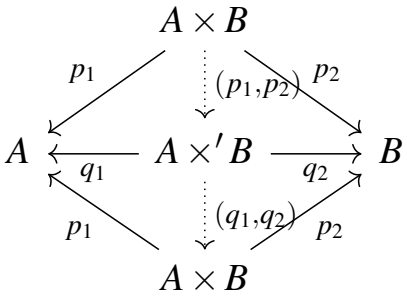
Proposição 3.4. *Em* ***Sets****, o produto entre dois objetos* $A, B \in Sets_0$ *é o produto cartesiano entre os dois conjuntos.*

Demonstração: Ora, para comprovar que o produto entre dois conjuntos é o produto cartesiano, é necessário, inicialmente, mostrar as projeções. Ora, basta definir $p : A \times B \to A$ como $p(a,b) = a$ e definir $q : A \times B \to B$ como $q(a,b) = b$. Agora é necessário provar a propriedade universal. Seja $f : Z \to A$ e $g : Z \to B$. Ora, basta definir uma $h : Z \to A \times B$ da seguinte forma: $h(x) = (f(x), g(x))$. Por fim, falta chegar que $p \circ h = f$ e $q \circ h = g$. Ora, $(p \circ h)(x) = p(h(x)) = p(f(x), g(x)) = f(x)$. Da mesma forma, $(q \circ h)(x) = q(h(x)) = q(f(x), g(x)) = g(x)$. Portanto, o produto cartesiano de dois produtos pode ser definitivamente considerado como o produto entre dois objetos de **Sets**. □

Seria possível encontrar uma outra definição aceitável para o produto de dois objetos em **Sets**? A resposta é encontrada na próxima proposição (AWODEY, 2010):

Proposição 3.5. *Se* $\times$ *e* $\times'$ *são produtos de uma categoria* $\mathbf{C}$*, então* $A \times B \cong A \times' B$*.*

Demonstração: A prova pode ser mais facilmente visualizável atraves do seguinte diagrama (AWODEY, 2010):



A chave para entender o funcionamento da prova é que, a partir da propriedade universal dos produtos, é possível obter as setas (p_1, p_2) e (q_1, q_2). Além disso, todos os triângulos do diagrama acima são comutativos. Olhando para o diagrama, as seguintes equações são obtidas a partir dos triângulos comutativos $(A \times B)(A \times B)A$ e $(A \times B)(A \times B)B$:

$$p_1 \circ (q_1, q_2) \circ (p_1, p_2) = p_1$$
$$p_2 \circ (q_1, q_2) \circ (p_1, p_2) = p_2$$

Ora, $(q_1, q_2) \circ (p_1, p_2)$ age como a seta identidade para $A \times B$. Ora, pela unicidade dessas setas, tem-se exatamente que $(q_1, q_2) \circ (p_1, p_2) = 1_{A \times B}$. Pela simetria, é possível obter que $(p_1, p_2) \circ (q_1, q_2) = 1_{A \times' B}$. Portanto, (p_1, p_2) e (q_1, q_2) são inversas e estabelecem o isomorfismo entre os dois produtos. □

Com essa prova, é possível afirmar que toda categoria tem apenas uma definição, em termos de estrutura, para o produto. Todas as outras possíveis serão estruturalmente semelhantes a essa única definição, já que são todas isomorfas entre si.

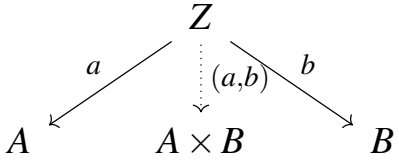
3.2.1 Produto na Teoria dos Tipos e das Categorias

Essa seção irá mostrar a estreita relação entre teoria dos tipos e teoria das categorias. Para isso, para cada construção da teoria dos tipos feita usando a dedução natural, será mostrada o diagrama ou a proposição equivalente na teoria das categorias. Naturalmente, a primeira dedução é a Formação do Produto $(\times - F)$ (NLAB, 2014):

$$\frac{A \text{ tipo} \qquad B \text{ tipo}}{A \times B \text{ tipo}} \times - F$$

A formação é intuitiva. Dado dois tipos A e B, é possível obter $A \times B$. Em categorias, há uma proposição equivalente: se $A, B \in C_0$, então $A \times B \in C_0$. Agora, a Introdução do Produto $(\times - I)$ (NLAB, 2014):

$$\frac{a : A \qquad b : B}{\langle a, b \rangle : A \times B} \times - I$$



Ora, dado um termo $a : A$ e outro $b : B$, é possível construir $\langle a, b \rangle : A \times B$. Nas categorias, isso é representado pelas setas a e b. Dados essas setas, é possível obter uma única $\langle a, b \rangle$ atraves

da propriedade universal dos produtos. A seguir, a Primeira Eliminação do Produto ($\times - E_1$) e a Segunda Eliminação do Produto ($\times - E_2$) (NLAB, 2014):

$$\frac{t : A \times B}{p_1(t) : A} \times -E_1 \qquad \frac{t : A \times B}{p_2(t) : B} \times -E_2$$

$$\begin{array}{ccccc} & & Z & & \\ & & \vdots\, t & & \\ A & \xleftarrow{p_1} & A \times B & \xrightarrow{p_2} & B \end{array}$$

As eliminações sao feitas com o auxílio das projeções. É exatamente o que representa o diagrama. Por fim, as regras de computação $p_1(\langle a,b\rangle) = a$ e o $p_2(\langle a,b\rangle) = b$ são representadas pelo diagrama completo:

$$\begin{array}{ccccc} & & Z & & \\ & \overset{a}{\swarrow} & \vdots\, \langle a,b\rangle & \overset{b}{\searrow} & \\ A & \xleftarrow{p_1} & A \times B & \xrightarrow{p_2} & B \end{array}$$

Com isso, foi possível, usando categorias, representar cada dedução natural e regra do produto. Usando o mesmo raciocínio, é possível traduzir qualquer entidade da teoria dos tipos para a teoria das categorias, incluindo o recursor J. Porém, devido a complexidade inerente do recursor J, a interpretação categórica também se torna demasiadamente complicada. Seria necessário definir diversos conceitos avançados que estão fora do escopo desse trabalho. Porém, é importante saber que realmente existe essa relação estreita entre essas duas teorias.

3.2.2 Coproduto

O objetivo dessa subseção não é detalhar o coproduto, mas mostrar como, através da dualidade, uma construção pode automaticamente gerar outra. Dado uma categoria **C** e objetos $A, B \in C_0$, o coproduto entre os dois objetos é denotado por $A + B$ e é representado pelo seguinte diagrama:

$$\begin{array}{ccccc} & & Z & & \\ & \overset{f}{\nearrow} & \vdots\, [a,b] & \overset{g}{\nwarrow} & \\ A & \xrightarrow[i_1]{} & A + B & \xleftarrow[i_2]{} & B \end{array}$$

Como pode-se ver, o coproduto é exatamente a categoria dual ao produto. As projeções são substituídas por injeções. Também tem uma propriedade universal, que diz que se existem $f : A \to Z$ e $g : B \to Z$, então existe apenas um $h : A + B \to Z$. A seguir, será mostrado como a dualidade pode ser utilizada para provar uma proposição:

Proposição 3.6. *Se $+$ e $+'$ são coprodutos de uma categoria **C**, então $A + B \cong A +' B$.*

Demonstração: A prova vem diretamente do fato que o duall de um isomorfismo é um isomorfismo. A dualidade só muda a direção de uma seta. Portanto, se uma seta é um isomorfismo na

categoria original, também o será na categoria oposta. Como o isomorfismo foi provado para o produto, basta pegar o diagrama oposto ao diagrama que foi utilizada na prova. O isomorfismo se manterá. □

Assim como no caso do produto, é possível estabelecer a relação da definição de coproduto da teoria dos tipos com o de categorias. Basta utilizar o mesmo processo utilizado para o produto.

3.3 Transformações Naturais

As transformações naturais são um dos conceitos mais importantes em teoria das categorias. Como já foi dito, foi um dos conceitos responsáveis pela própria criação dessa teoria. Uma ótima forma de começar é com o exemplo que, se A, B e C são objetos de uma categoria que possui o produto, então $(A \times B) \times C \cong A \times (B \times C)$ (AWODEY, 2010). A prova desse isomorfismo será omitida, já que é um pouco longa e não adiciona nenhuma informação útil. Além disso, é bastante simples, basta desenhar os diagramas e procurar estabelecer o isomorfismo através das comutatividades. De qualquer forma, tem-se um isomorfismo $f : (A \times B) \times C \xrightarrow{\sim} A \times (B \times C)$. O isomorfismo f tem uma grande limitação. Apenas estabelece o isomorfismo para os objetos A, B e C. E se A fosse substituído por um A'? O isomorfismo se manteria? Como é possível provar que o isomorfismo independe das escolhas dos objetos? Para isso, é necessário usar transformações naturais.

Intuitivamente, uma transformação natural é um morfismo entre funtores. Ou seja, dados funtores F e G, uma transformação natural é um $\theta : F \to G$. O uso de transformações naturais fica evidente no exemplo da associatividade do produto. Primeiro, pode-se pensar nos seguintes funtores: $F : \mathbf{C} \times \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ definido por $F = (-_1 \times -_2) \times -_3$ e o funtor $G : \mathbf{C} \times \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ definido por $G = -_1 \times (-_2 \times -_3)$. Com isso, seria preciso achar uma transformação natural $\theta : F \to G$. Além disso, seria necessário provar que θ é um isomorfismo, gerando uma entidade conhecida como isomorfismo natural. Só assim seria provado que o isomorfismo $(A \times B) \times C \cong A \times (B \times C)$ independe da escolha dos objetos. Nesse caso, é dito que o isomorfismo é natural em A, B e C. A seguir, a definição formal de transformação natural:

Definição 3.13 (Transformação natural (AWODEY, 2010))**.** *Dado categorias $\mathbf{C}$ e $\mathbf{D}$ e funtores $F, G : \mathbf{C} \to \mathbf{D}$, uma transformação natural $\theta : F \to G$ é uma família de setas em $\mathbf{D}$ dada por: $(\theta_C : FC \to GC)_{C \in C_0}$. Além disso, dado um $f : C \to C' \in C_1$, então $\theta_{C'} \circ F(f) = G(f) \circ \theta_C$. Essa igualdade é representada pelo seguinte quadrado comutativo:*

$$\begin{array}{ccc} FC & \xrightarrow{\theta_C} & GB \\ \downarrow{\scriptstyle F(f)} & & \downarrow{\scriptstyle G(g)} \\ FC' & \xrightarrow[\theta_{C'}]{} & GD \end{array}$$

As setas θ_C são chamadas de componente θ em C.

Olhando para a definição acima, é possível entender que achar uma transformação natural entre dois funtores se resume a achar um morfismo adequado para os componentes. Esse morfismo precisa respeitar a naturalidade, ou seja, tem que obedecer o quadrado comutativo. A partir da definição de transformação natural, é possível definir a seguinte categoria:

Definição 3.14 (Categoria $Fun(\mathbf{C},\mathbf{D})$ (AWODEY, 2010))**.** *Dado categorias $\boldsymbol{C}$ e $\boldsymbol{D}$, é possível construir a categoria $Fun(\boldsymbol{C},\boldsymbol{D})$ definida por:*

- *Objetos: os objetos são funtores $F : \boldsymbol{C} \to \boldsymbol{D}$.*
- *Setas: as setas são transformações naturais $\theta : F \to G$.*
- *Composição: Dado setas $\theta : F \to G$ e $\phi : G \to H$, é possível definir $(\phi \circ \theta)$ a partir das componentes: $(\phi \circ \theta)_C = \phi_C \circ \theta_C$.*
- *Identidade: Dado um funtor F, a seta 1_F é definida a partir das componentes: $(1_F)_C = 1_{FC} : FC \to FC$.*

Com isso, é finalmente possível definir um isomorfismo natural:

Definição 3.15 (Isomorfismo natural (AWODEY, 2010))**.** *Um isomorfismo natural é uma transformação natural $\theta : F \to G$ que é um isomorfismo na categoria $Fun(\boldsymbol{C},\boldsymbol{D})$.*

Existe uma forma mais simples de verificar se uma transformação natural é um isomorfismo (AWODEY, 2010):

Proposição 3.7. *Uma transformação natural $\theta : F \to G$ é um isomorfismo natural se e somente se cada componente $\theta_C : FC \to GC$ é um isomorfismo.*

Demonstração: ($\Rightarrow$) Começando com a ida, é necessário provar que, dado um isomorfismo natural $\theta : F \to G$, então toda componente θ_C também é um isomorfismo. Já que θ é um isomorfismo, então existe um $\phi : G \to F$ que é inverso a θ. Dessa forma, tem-se que $\theta \circ \phi = 1_G$ e $\phi \circ \theta = 1_F$. Portanto, pegando uma componente em C, tem-se que $(\theta \circ \phi)_C = (1_G)_C$ e $(\phi \circ \theta)_C = (1_F)_C$. Pela definição de composição e identidade em Fun, obtém-se que $\phi_C \circ \theta_C = 1_{FC}$ e $\theta_C \circ \phi_C = 1_{GC}$. Com isso, conclui-se que ϕ_C é inversa a θ_C. Logo, cada componente é um isomorfismo.

($\Leftarrow$) Para a volta, é necessário provar que se toda componente θ_C de uma transformação natural θ é um isomorfismo, então θ também o é. Se cada componente é um isomorfismo, então para cada θ_C existe uma seta inversa ϕ_C tal que $\theta_C \circ \phi_C = 1_{GC}$ e $\phi_C \circ \theta_C = 1_{FC}$. Pela definição de composição e identidade de Fun, tem-se que $\theta_C \circ \phi_C = (\theta \circ \phi)_C = 1_{GC} = (1_G)_C$. Logo, $\theta \circ \phi = 1_G$. Analogamente, prova-se que $\phi \circ \theta = 1_F$. Portanto, ϕ é a transformação inversa a θ. Conclui-se que θ é uma transformação natural. □

A proposição acima torna mais simples trabalhar com isomorfismos naturais, já que basta provar que cada componente é um isomorfismo. Para mostrar o uso na prática de isomorfismos e transformações naturais, pode-se pensar na prova da seguinte proposição:

Proposição 3.8. *Seja* $\mathbf{C}$ *uma categoria que tem produtos binários e* $A, B \in C_0$*, então* $A \times B \cong B \times A$ *e o isomorfismo é natural em* A *e* B*.*

Se a demonstração se resumisse a provar que $A \times B \cong B \times A$, então seria apenas necessário usar os diagramas do produto e estabelecer o isomorfismo. Porém, como é necessário provar a naturalidade, a única forma é usando transformações naturais. Segue a prova (AWODEY, 2010):

Demonstração: Para provar o isomorfismo natural, primeiro é necessário definir os funtores. O primeiro funtor é o $\times : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, definido por $\times = (-_1, -_2)$. O segundo funtor é $\bar{\times} : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ definido por $\bar{\times} : (-_2, -_1)$. Ou seja, em objetos, $A \bar{\times} B = B \times A$ e em setas, $\alpha \bar{\times} \beta = \beta \times \alpha$. Com isso, é necessário agora pensar em uma transformação natural θ entre os dois funtores. Pode-se definir θ a partir de sua ação em termos (A, B). Dessa forma, pode-se definir como $\theta_{(A,B)}(a,b) = (b,a)$, com a e b sendo setas de um produto (Ou seja, dado um Z, são as setas $Z \to A$ e $Z \to B$ respectivamente). É necessário provar que θ é uma transformação natural. Em outras palavras, demonstar que o seguinte diagrama comuta:

$$\begin{array}{ccc} A \times B & \xrightarrow{\theta_{(A,B)}} & B \times A \\ \downarrow{\scriptstyle \alpha \times \beta} & & \downarrow{\scriptstyle \beta \times \alpha} \\ A' \times B' & \xrightarrow[\theta_{(A',B')}]{} & B' \times A' \end{array}$$

Para provar que o diagrama comuta, basta considerar um (a,b) qualquer originado de um Z, sendo $a : Z \to A$ e $b : Z \to B$. Com isso, as seguintes equações são obtidas:

$$\begin{gathered} (\beta \times \alpha)\theta_{(A,B)}(a,b) = (\beta \times \alpha)(b,a) \\ (\beta \times \alpha)(b,a) = (\beta b, \alpha a) \\ (\beta b, \alpha a) = \theta_{(A',B')}(\alpha a, \beta b) \\ (\alpha a, \beta b) = \theta_{(A',B')}(\alpha \times \beta)(a,b) \end{gathered}$$

Portanto, $(\beta \times \alpha)\theta_{(A,B)}(a,b) = \theta_{(A',B')}(\alpha \times \beta)(a,b)$, comprovando que o quadrado é comutativo. Logo, conclui-se que θ é realmente uma transformação natural. Porém, ainda falta provar que θ é um isomorfismo. Para isso, pode-se usar a **proposição 3.7**, ou seja, basta provar que cada componente é um isomorfismo. O isomorfismo para cada componente é simples de provar, já que para cada $\theta_{(A,B)}$ tem-se a inversa $\phi_{(A,B)} = \theta_{(B,A)}$. Segue a checagem:

$$\begin{gathered} (\theta_{(A,B)} \circ \phi_{(A,B)})(B \times A) = \theta_{(A,B)}(A \times B) = B \times A \\ (\phi_{(A,B)} \circ \theta_{(A,B)})(A \times B) = \phi_{(B,A)}(B \times A) = A \times B \end{gathered}$$

Portanto, conclui-se que:

$$(\theta_{(A,B)} \circ \phi_{(A,B)}) = 1_{\bar{\times}(A,B)}$$
$$(\phi_{(A,B)} \circ \theta_{(A,B)}) = 1_{\times(A,B)}.$$

Portanto, cada componente θ em (A,B) é um isomorfismo. Logo, θ é um isomorfismo, concluindo, assim, a prova. □

3.4 Considerações finais

Nesse capítulo foram vistos os principais conceitos de teorias de categorias. Todos são de grande importância, porém, os conceitos de isomorfismo e grupóide terão uma importância especial, já que serão a base para a interpretação que será feita no próximo capítulo. Outros conceitos, como a de transformação e isomorfismo naturais, serão utilizados para definir categorias mais complicadas, como as de alta ordem. Além disso, a partir do exemplo do produto entre objetos de uma categoria, foi mostrado que a teoria dos tipos está intrinsecamente conectada com a teoria das categorias.

Para não alongar desnecessariamente o capítulo, alguns conceitos interessantes, infelizmente, tiveram que ser omitidos. São conceitos como limite e colimite de categorias, equivalência e adjuntores. Felizmente, porém, esses conceitos não serão importantes para desenvolver os capítulos 4 e 5.

Por fim, vale ressaltar que nem todos os conceitos que serão necessários para entender o capítulo 5 foram mostrados nesse capítulo. O capítulo 5 focará em um tipo mais complicado de categoria, conhecida como categoria de alta ordem. Como as categorias de alta ordem são entidades bastante complexas, juntar esses conceitos mais avançados com os conceitos desse capítulo, do ponto de vista didático, seria uma escolha insensata. Portanto, as categorias de alta ordem serão devidamente introduzidas e definidas no capítulo 5. Por sua vez, todos os conceitos necessários para desenvolver o próximo capítulo foram introduzidos no presente capítulo.

4

Teoria das Categorias e Caminhos Computacionais

Esse capítulo tem como objetivo principal mostrar como os caminhos computacionais podem ser interpretados a partir dos conceitos da teoria das categorias. Ou seja, estudar as propriedades matemáticas dos caminhos computacionais utilizando a teoria das categorias. No capítulo passado, foi visto que uma categoria obedece regras de identidade e associatividade. Já no *capítulo 2*, foi mostrado que os caminhos computacionais seguem as regras de reflexividade, transitividade e simetria. Esse capítulo mostrará que essas regras têm uma conexão direta com a associatividade e a identidade das categorias. Além disso, será mostrado que uma categoria pode ser considerada estrita ou fraca, dependendo de como as igualdades se estabelecem. O objetivo desse capítulo é, então, preencher a falta de interpretação matemática para os caminhos computacionais. O uso de uma teoria bem estabelecida da matemática, a teoria das categorias, para interpretar os caminhos computacionais adicionam um importante significado matemático a essa entidade. A esperança é que ao juntar o significado matemático com as claras vantagens computacionais dos caminhos, essa entidade se tornará cada vez mais utilizada como forma padrão de definir o tipo identidade. Alcançando isso, a teoria dos tipos tornar-se-ia muito mais simples e acessível.

Para obter os resultados necessários, será introduzido um sistema de reescritas entre caminhos computacionais. Será mostrado quais as condições necessárias para um caminho se reduzir ao outro. Além do mais, a ideia que existe um caminho que conecta caminhos será introduzida. De fato, essa interpretação irá ainda mais além: se existe um caminho que conecta caminhos, então existirá um caminho entre caminhos que conectam caminhos. Ora, será visto que esse processo se estende infinitamente. Será exatamente esse raciocínio que motivará o uso de categorias de alta ordem, as quais serão desenvolvidas no capítulo 5.

Outro ponto importante é que essa interpretação mostrará que importantes resultados do tipo identidade sem caminhos também podem ser obtidos através do uso de caminhos computacionais. A partir do tipo identidade sem caminhos, é possível induzir uma estrutura de grupóide para qualquer tipo da teoria dos tipos. A interpretação categórica dos caminhos

mostrará que essa estrutura também pode ser induzida a partir dos caminhos computacionais. De fato, se o objetivo é substituir o tipo identidade para uma abordagem com caminhos, todos os resultados obtidos através da abordagem sem caminhos devem ser preservados. Ou seja, a introdução dos caminhos computacionais deve abrir caminho para se provar todos os resultados já estabelecidos. Boa parte dessas provas não serão feitas nesse trabalho, já que dependem de conceitos extremamente complexos que estão fora do escopo (isso será explicado melhor no capítulo 5). Porém, a interpretação que será dada pode ser entendida como o esforço inicial, a base necessária para desenvolver resultados mais complicados no futuro.

4.1 Sistema de Reescrita de Caminhos

Essa seção falará sobre um sistema que formaliza a ideia que existem caminhos entre os próprios caminhos computacionais. Porém, antes de explicar como funciona essa reescrita de caminhos, é necessário introduzir três caminhos essenciais (QUEIROZ; OLIVEIRA, 2011):

$$\frac{a =_t b : A \qquad b =_u c : A}{a =_{\tau(t,u)} c : A}\ \textit{transitividade} \qquad \frac{a : A}{a =_\rho a : A}\ \textit{reflexividade}$$

$$\frac{a =_t b : A}{b =_{\sigma(t)} a : A}\ \textit{simetria}$$

As propriedades são obtidas diretamente através dos axiomas. A transitividade a partir do axioma τ, a simetria do axioma σ e a reflexividade do ρ. Com esses caminhos, já é possível entender a ideia de caminhos entre caminhos.

4.1.1 O Sistema de Reescrita de Caminhos

Talvez a melhor forma de explicar a ideia que um caminho pode ser reescrito, originando um outro caminho, seja através dos seguinte exemplo:

Exemplo 4.1. *Considere um caminho $a =_t b : A$. É possível aplicar a propriedade simétrica a esse caminho, obtendo-se, então, $b =_{\sigma(t)} a : A$. É possivel, então, aplicar novamente a simetria, obtendo-se $a =_{\sigma(\sigma(t))} b : A$. Ou seja, a partir de um caminho t, a simetria foi aplicada duas vezes em seqûencia, obtendo um caminho equivalente ao caminho t. Ora, o caminho $\sigma(\sigma(t))$ é apenas uma forma redudante de expressar o caminho t. Porém, mesmo um sendo uma forma redudante do outro, os caminhos são diferentes pela definição de caminhos computacionais. Seria, então, natural pensar que ao menos o caminho $\sigma(\sigma(t))$ deveria poder ser reduzido (ou reescrito) ao caminho t.*

O exemplo envolvendo a simetria é um dos mais simples. A seguir, exemplos envolvendo redundâncias geradas pela transitividade e reflexividade:

Exemplo 4.2. *Considere o caminho reflexivo $a =_\rho a : A$. É possível, então, aplicar a simetria, obtendo-se $a =_{\sigma(\rho)} a : A$. Ora, o caminho obtido é equivalente ao inicial, já que a simetria foi aplicada ao caminho reflexivo. Portanto, $\sigma(\rho)$ é uma forma redundante de expressar o caminho ρ. Desas forma, $\sigma(\rho)$ deveria reduzir-se a ρ.*

Exemplo 4.3. *Considere um caminho $a =_t b : A$. Aplicando a propriedade simétrica, obtém-se o caminho $b =_{\sigma(t)} a : A$. Agora, é possível pegar os dois caminhos e aplicar a transitividade, obtendo-se $a =_{\tau(t,\sigma(t))} a : A$. Como um caminho é inverso ao outro, a aplicação da transitividade origina um caminho equivalente ao caminho reflexivo. Dessa forma, o caminho $\tau(t,\sigma(t))$ deveria se reduzir ao caminho reflexivo, ou seja, ρ.*

Como pode-se notar nos exemplos, caminhos diferentes devem ser considerados iguais se um caminho é apenas uma forma redundante de um outro. Os examplos que acabaram de ser dados são apenas casos simples e diretos. Entretanto, existem exemplos mais complexos e difíceis de serem identificados. Foi visto que a teoria da igualdade para a teoria dos tipos tem um total de sete axiomas, dessa forma, o número de possíveis combinações e redundâncias geradas é bastante alto. Felizmente, o mapeamento de todas as redundâncias foi minuciosamente feito por OLIVEIRA (1995). Nesse trabalho, de Oliveira propõe um sistema que mapeia e identifica todas as possíveis redundâncias. Esse sistema, conhecido como Sistema de Reescrita de Termos ($LND_{EQ} - TRS$), tem um total de 39 regras que removem redundâncias. Para a interpretação categórica, apenas as regras envolvendo os axiomas de transitividade, reflexividade e simetria serão importantes. Por isso, não serão mostradas todas as regras do $LND_{EQ} - TRS$, mas apenas um subconjunto contendo as regras necessárias para esse trabalho. Essas regras são todas provenientes de OLIVEIRA (1995); QUEIROZ; OLIVEIRA (2011):

- Regras envolvendo σ e ρ

$$\frac{x =_\rho x : A}{x =_{\sigma(\rho)} x : A} \quad \rhd_{sr} \quad x =_\rho x : A$$

$$\frac{\dfrac{x =_r y : A}{y =_{\sigma(r)} x : A}}{x =_{\sigma(\sigma(r))} y : A} \quad \rhd_{ss} \quad x =_r y : A$$

Dessas reduções, as seguintes regras são obtidas:

$\sigma(\rho) \rhd_{sr} \rho$

$\sigma(\sigma(r)) \rhd_{ss} r$

- Regras envolvendo τ

$$\frac{x =_r y : A \qquad y =_{\sigma(r)} x : A}{x =_{\tau(r,\sigma(r))} x : A} \quad \rhd_{tr} \quad x =_\rho x : A$$

$$\frac{y =_{\sigma(r)} x : A \qquad x =_r y : A}{y =_{\tau(\sigma(r),r)} y : A} \quad \rhd_{tsr} \quad y =_\rho y : A$$

$$\frac{x =_r y : A \qquad y =_\rho y : A}{x =_{\tau(r,\rho)} y : A} \quad \rhd_{trr} \quad x =_r y : A$$

$$\frac{x =_\rho x : A \qquad x =_r y : A}{x =_{\tau(\rho,r)} y : A} \quad \rhd_{tlr} \quad x =_r y : A$$

Regras obtidas:

$\tau(r,\sigma(r)) \rhd_{tr} \rho$

$\tau(\sigma(r),r) \rhd_{tsr} \rho$

$\tau(r,\rho) \rhd_{trr} r$

$\tau(\rho,r) \rhd_{tlr} r$

- Regras envolvendo τ and τ

$$\frac{\dfrac{x =_t y : A \qquad y =_r w : A}{x =_{\tau(t,r)} w : A} \qquad w =_s z : A}{x =_{\tau(\tau(t,r),s)} z : A}$$

$$\rhd_{tt} \frac{x =_t y : A \qquad \dfrac{y =_r w : A \qquad w =_s z : A}{y =_{\tau(r,s)} z : A}}{x =_{\tau(t,\tau(r,s))} z : A}$$

Regra obtida:

$\tau(\tau(t,r),s) \rhd_{tt} \tau(t,\tau(r,s))$

Agora que foi introduzido esse sistema que remove redundâncias causadas pelos caminhos computacionais, pode-se definir alguns conceitos importantes:

Definição 4.1 (Regra *rw*)**.** *Uma regra rw é qualquer uma das regras encontradas em* $LND_{EQ} - TRS$.

Definição 4.2 (*rw*-contração e *rw*-redução)**.** *Sejam s e t caminhos computacionais. É dito que* $s \rhd_{1rw} t$ *(lido como: s rw-contrai para t) sse é possível obter t a partir da aplicação de apenas uma regra rw em s. Se for possível reduzir s a t através de uma aplicação de um número finito de regras rw, então é dito que* $s \rhd_{rw} t$ *(lido como: s rw-reduz para t).*

Definição 4.3 (*rw*-igualdade)**.** *Sejam s e t caminhos computacionais. É dito que $s =_{rw} t$ (lido como: s é rw-igual a t) sse é possível obter t a partir de s através de um número finito (ou até mesmo 0) de rw-contrações e rw-contrações inversas. Em outras palavras, $s =_{rw} t$ sse existe uma sequência $R_0, ..., R_n$, com $n \geq 0$ tal que:*

$$(\forall i \leq n-1)(R_i \rhd_{1rw} R_{i+1} \text{ ou } R_{i+1} \rhd_{1rw} R_i)$$
$$R_0 \equiv s, \quad R_n \equiv t$$

A partir dessas definições, é possível concluir que:

Proposição 4.1. *rw-igualdade é transitiva, reflexiva e simétrica.*

Demonstração: A proposição é uma consequência direta do fato que a *rw*-igualdade foi definida como o fecho simétrico, reflexivo e transitivo da *rw*-redução. □

O fato que a *rw*-igualdade é transitiva, reflexiva e simétrica é de grande importância, já que torna a *rw*-igualdade em uma classe de equivalência. Com isso, já é possível entender a relação entre caminhos computacionais e teoria das categorias. Porém, antes, é importante mencionar que o sistema $LND_{EQ} - TRS$ tem duas propriedades importantes: é confluente e sempre termina. Sempre termina, pois para qualquer caminho s, é possível aplicar sucessivamente regras *rw* até que obtenha-se um caminho sem redundâncias. É confluente, pois dado que $s \rhd_{rw} t$ e $s \rhd_{rw} w$, então é possível obter um z tal que $t \rhd_{rw} z$ e $w \rhd_{rw} z$. As provas desses fatos são encontradas em OLIVEIRA (1995); OLIVEIRA; QUEIROZ (1994, 1999); QUEIROZ; OLIVEIRA; GABBAY (2011).

4.2 A Interpretação Categórica - O Modelo de Grupóide

O objetivo dessa seção é finalmente mostrar a ligação entre caminhos computacioanais e a teoria das categorias. A ideia é que um caminho computacional tem claramente uma interpretação categórica, podendo ser visto como um morfismo de uma categoria formada por termos de um determinado tipo. Além disso, essa categoria induzida pelo caminho computacional seria uma grupóide. Como já foi visto que um caminho computacional é basicamente um termo do tipo identidade, então o fato que os caminhos induzem uma grupóide implicaria diretamente que o tipo identidade é uma estrutura de grupóide. De fato, foi provado por HOFMANN; STREICHER (1994) que a abordagem tradicional do tipo identidade (sem caminhos computacionais) induz uma estrutura de grupóide. Então, dado que esse trabalho propõe que o tipo identidade seja modelado a partir de caminhos computacionais, é natural pensar que essa nova abordagem deve ser capaz de obter o mesmo resultado. É exatamente isso que será feito nessa seção.

4.2.1 O Modelo de Grupóide

Foi provado por HOFMANN; STREICHER (1994) que o tipo identidade sem caminhos computacionais induzem uma estrutura de grupóide. Para isso, foram construídos termos para os

seguintes tipos (HOFMANN; STREICHER, 1994):

- $Id_{Id_A(x,z)}(trans(trans(p,q),r),trans(p,trans(q,r)))$
- $Id_{Id_A(x,y)}(trans(refl(x),r),r)$
- $Id_{Id_A(x,y)}(trans(r,refl(x)),r)$
- $Id_{Id_A(y,y)}(trans(symm(r),r),refl(x))$
- $Id_{Id_A(x,x)}(trans(r,symm(r)),refl(x))$
- $Id_{Id_A(x,y)}(symm(symm(r)),r)$

Ao provar que cada um desses tipos é habitado, HOFMANN; STREICHER (1994) prova que o tipo identidade realmente induz um modelo de grupóide. Porém, como essas igualdades são proposicionais, elas não se sustentam sem evidência. Elas apenas se sustentam a nível proposicional. Por isso, é dito que essa estrutura é *fraca*. Um problema da abordagem de HOFMANN; STREICHER (1994) é que não foi exposta explicitamente a origem dos identificadores *trans* e *symm*, pois somente o identificador $refl$ faz parte da definição do tipo identidade tal qual originalmente formulado por Martin-Löf.

A abordagem que será utilizada nesse trabalho para provar que caminhos computacionais também induzem uma estrutura de grupóide será diferente da vista acima. A principal diferença é que não serão feitas construções de nenhum tipo. De fato, não será utilizado o tipo identidade e nenhuma regra de eliminação. Será visto, utilizando inteiramente conceitos de teoria das categorias, que o caminho computacional naturalmente induz uma categoria que tem uma estrutura de grupóide. Nesse caso, será utilizada a definição categórica de grupóide, ou seja, que cada seta será um isomorfismo.

4.2.2 A Interpretação: a Grupóide Induzida

Antes de mostrar a categoria induzida por caminhos computacionais, é importante deixar claro que a estrutura induzida será uma estrutura *fraca*. O termo *fraco* será utilizado para indicar que as igualdades da estrutura de categoria (ou seja, as igualdades da lei da associatividade e da identidade) não irão se sustentar no sentido estrito. Irão apenas se sustentar a um outro nível de abstração. No caso da grupóide de HOFMANN; STREICHER (1994), foi visto que esse nível de abstração é na verdade a igualdade proposicional. No presente trabalho, esse novo nível de abstração será a *rw*-igualdade.

Dado um tipo qualquer A, é possível induzir, a partir dos caminhos computacionais, a seguinte estrutura:

Definição 4.4 (Estrutura induzida A_{rw})**.** *A estrutura induzida A_{rw} é a seguinte estrutura:*

- *Objetos: os objetos são termos a do tipo A. Em outras palavras, são $a : A$.*

- *Morfismos: os morfismos (ou setas) são caminhos computacionais s entre dois objetos* $a,b:A$. *Em outras palavras, existe uma seta* $s:a\to b$ *se* $a=_s b:A$.

Proposição 4.2. *A estrutura* A_{rw} *é uma categoria fraca.*

Demonstração: Para mostrar que é uma categoria, precisa-se definir a composição entre morfismos, mostrar o morfismo identidade e checar as leis de associatividade e de identidade. Como se está trabalhando com uma estrutura fraca, as leis precisam apenas se sustentar a nível de *rw*-igualdade.

- Composições: Dado setas (caminhos) $s:a\to b$ e $r:b\to c$, é preciso achar uma seta $t:a\to c$ tal que $t=r\circ s$. Para isso, é preciso definir o significado de uma composição entre caminhos. Já que a igualdade tem o axioma da transitividade, é natural definir uma composição como uma aplicação da transitividade. Ou seja, $t=r\circ s=\tau(s,r)$. Dessa forma, dado setas $s:a\to b$ e $r:b\to c$, sempre é possível obter $t:a\to c=r\circ s=\tau(s,r)$.

- Associatividade da composição: Dado setas $s:a\to b$, $r:b\to c$ e $t:c\to d$, já que se está trabalhando com uma entidade fraca, e necessário concluir que $t\circ(r\circ s)=_{rw}(t\circ r)\circ s$. Em outras palavras, é necessário concluir que $\tau(\tau(s,r),t)=_{rw}\tau(s,\tau(r,t))$. Ora, isso é uma consequência direta da seguinte equação:

$$tt:\tau(\tau(s,r),t)\rhd_{tt}\tau(s,\tau(r,t))$$

  Portanto, a lei da associatividade é satisfeita fracamente, a nível de *rw*-igualdade.

- Flecha da identidade: para qualquer objeto a, considere o caminho reflexivo $a=_\rho a$ como a seta da identidade 1_a. Essa seta identidade é chamada de ρ_a, para indicar que é o caminho reflexivo de a;

- Lei da identidade: Para qualquer seta $s:a\to b$, é necessario mostrar que $s\circ 1_a=s=1_b\circ s$. Essas igualdades podem ser mostradas, no sentido fraco, com a aplicação direta de algumas regras *rw*:

$$s\circ 1_a=s\circ\rho_a=\tau(\rho_a,s)\rhd_{tlr}s$$
$$1_b\circ s=\rho_b\circ s=\tau(s,\rho_b)\rhd_{trr}s$$

  Logo, $s\circ 1_a=_{rw}s=_{rw}1_b\circ s$

Dessa forma, todas as condições necessárias para A_{rw} ser uma categoria (fraca) foram estabelecidas. Portanto, conclui-se que A_{rw} é uma categoria fraca.

□

Agora que foi concluido que A_{rw} é uma categoria fraca, é possível mostrar que A_{rw} é uma grupóide fraca:

Proposição 4.3. *A_{rw} é uma grupóide fraca.*

Demonstração: Para provar que é uma grupóide, é necessário mostrar que cada seta é um isomorfismo. Já que se está trabalhando com uma estrutura fraca, os isomorfismos apenas precisam se sustentar a nível de *rw*-igualdade. Portanto, dado uma seta qualquer $s : a \to b$, é necessário achar uma t tal que $s \circ t =_{rw} 1_b$ e $t \circ s =_{rw} 1_a$. Ora, já é sabido que, dado um caminho s, basta aplicar o axioma σ para obter o caminho inverso $\sigma(s)$. Portanto, basta fazer $t = \sigma(s)$. As seguintes equações justificam a *rw*-igualdade:

$$s \circ t = s \circ \sigma(s) = \tau(\sigma(s), s) \rhd_{tsr} \rho_b.$$
$$t \circ s = \sigma(s) \circ s = \tau(s, \sigma(s)) \rhd_{tr} \rho_a$$

Portanto, $\sigma(t)$ é realmente a seta inversa a s (no sentido fraca). Portanto, cada seta é um isomorfismo fraco e, então, a estrutura é uma grupóide fraca.

□

Com isso, conclui-se que caminhos computacionais são realmente capazes de induzir uma estrutura de grupóide. Uma questão interessante é se seria possível definir uma estrutura estrita, ao invés de uma estrutura fraca. A resposta é **sim**. Para isso, as igualdades teriam que se sustentar realmente, sem ser utilizado nenhum nível de abstração. Como a *rw*-igualdade é transitiva, simétrica e reflexiva, a *rw*-igualdade pode, então, ser considerada uma classe de equivalência. Dessa forma, se $s =_{rw} t$, então s e t seriam da mesma classe de equivalência. Em outras palavras, ter-se-ia que $[s]_{rw} = [r]_{rw}$.. Portanto, a partir da estrutura A_{rw}, é possível obter uma estrutura estrita $[A_{rw}]$. Para isso, a única mudança necessária é que, ao invés de considerar setas como caminhos $s : a \to b$, bastaria considerar que uma seta é a classe de equivalência de um caminho. Ou seja, $[s]_{rw} : a \to b$ é uma seta de A_{rw} módulo *rw*-igualdade. Dessa forma, definindo a composição como $[s]_{rw} \circ [r]_{rw} = [s \circ r]_{rw}$ e a seta identidade como $[\rho_a]_{rw}$, as igualdades se manterão de forma estrita. Isso se dá pelo fato de que as igualdades são as mesmas da *proposição 4.2*, mas com a diferença que as classes de equivalência que estão sendo consideradas. Da mesma forma, a igualdade da grupóide se sustentará de uma forma estrita. Portanto, $[A_{rw}]$ é uma grupóide estrita. Com isso, conclui-se os principais resultados desse capítulo.

4.3 Considerações Finais Sobre a Interpretação Categórica

Nesse capítulo, foi visto que é possível encontrar redundâncias dentro de um caminho computacional. Além disso, essas redundâncias podem ser resolvidas, gerando caminhos sem redundâncias. Também foi visto que um sistema que as resolve já foi previamente proposto por OLIVEIRA (1995); QUEIROZ; OLIVEIRA (2011). Sendo um sistema bastante complexo, com um total de 39 regras, foi concluído que apenas um pequeno subconjunto de regras eram necessárias para obter os resultados envolvendo categorias. As regras usadas foram as envolvendo redundâncias da transitividade, reflexividade e simetria. Utilizando essas regras, foi visto que é

possível, a partir dos caminhos computacionais, obter uma estrutura de categoria em um sentido fraco, pois as igualdades apenas se sustentam a nível de um conceito definido como *rw*-igualdade. Por fim, foi concluído que essa estrutura de categoria tem as propriedades de grupóide.

A grande importância desse capítulo foi mostrar as propriedades matemáticas dos caminhos computacionais. Definido originalmente como uma estrutura computacional da teoria da igualdade, as propriedades puramente matemáticas dessa entidade ainda não tinham sido anteriormente exploradas. Com o resultado desse capítulo, foi mostrado que os caminhos computacionais realmente tem propriedades importantes, como a indução de um modelo de grupóide. Esses resultados ganham mais importância devido ao fato que, como visto no capítulo 2, é possível usar os caminhos computacionais para desenvolver o tipo identidade, já que esse tipo seria apenas o tipo de um caminho computacional. Como é sabido que a abordagem clássica do tipo identidade segue um modelo de grupóide (HOFMANN; STREICHER, 1994), mostrar o mesmo fato para a abordagem de caminhos introduzida no presente trabalho era um ponto essencial. Isso foi feito no presente capítulo, com a vantagem que a modelagem surge naturalmente das propriedades dos caminhos, sem ser necessário o uso das regras de eliminação e da construção de termos da teoria dos tipos.

O próximo capítulo irá levar essa interpretação para um novo nível de abstração, passando a se utilizar categorias de alta ordem.

5

Caminhos Computacionais e Categorias de Alta Ordem

No capítulo passado, foi visto que existe uma interpretação do tipo identidade formulado com identificadores para caminhos computacionais na teoria das categorias. Além disso, foi provado que o tipo identidade com caminhos computacionais induz uma estrutura importante conhecida como grupóide. O objetivo desse capítulo é expandir esses resultados, utilizando estruturas de alta ordem conhecidas como categorias de alta ordem (do inglês, *higher categories*). A ideia é que a existência de caminhos entre caminhos estabelecidos pela *rw*-igualdade cria uma estrutura multidimensional com níveis de abstração maiores que o de uma categoria normal. O objetivo é, então, utilizar as categorias de alta ordem para modelar essa estrutura multidimensional.

A construção da estrutura multidimensional será minuciosamente explicada. Porém, antes disso, é necessário introduzir conceitos novos sobre a chamada teoria das categorias de alta ordem. Esses conceitos serão definidos a partir dos conceitos mostrados no *capítulo 3*, mas com a diferença que serão conceitos mais avançados, já que envolvem estruturas de vários níveis de dimensão.

Na primeira parte será mostrada uma extensão do sistema $LND_{EQ} - TRS$. Essa extensão será baseada no fato de que é possível encontrar redundâncias entre *rw*-igualdades, sendo necessário criar um novo sistema que as resolva. Na segunda parte do capítulo, serão definidos os conceitos de conjuntos globulares, o conceito de composição horizontal e o de uma bicategoria. Na terceira parte, serão utilizados os conceitos desenvolvidos na primeira e segunda parte para induzir uma categoria de alta ordem de 2 dimensões a partir dos caminhos computacionais. Por fim, serão feitas conjecturas envolvendo estruturas induzidas com mais que 2 dimensões. Além disso, a ultima parte montará a base para trabalhos e pesquisas futuros.

Os resultados desse capítulo são baseados em artigo de autoria do próprio, juntamente com de Queiroz e de Oliveira. O artigo foi preliminarmente aceito na conferência LSFA - 2015. Link do arxiv: http://arxiv.org/pdf/1504.04759v1.pdf (RAMOS; QUEIROZ; OLIVEIRA, 2015b)

5.1 Os Múltiplos $LND_{EQ} - TRS$

No capítulo passado, foi concluído que existe um sistema com um conjunto de regras que resolvem as redundâncias presentes em um caminho computacional. Além disso, foi dito que essas redundâncias são geradas por utilzações redundantes dos axiomas da igualdade. É dito que esse sistema, conhecido como $LND_{EQ} - TRS$, resolve as redundâncias causadas pela igualdade entre termos da teoria dos tipos. Além disso, foi dito que esse sistema tem um total de 39 regras. Porém, se a igualdade causa redundâncias, a mesma ideia poderia ser estendida para a *rw*-igualdade. Se apenas 7 axiomas são reponsáveis por 39 redundâncias e regras, imagine as redundâncias possivelmente causadas por essas 39 regras da *rw*-igualdade. De fato, pode-se pensar em um sistema que mapeie e resolva todas as redundâncias causadas pela *rw*-igualdade. Esse sistema será chamado de $LND_{EQ} - TRS_2$. Um trabalho minucioso que estuda todas as possíveis regras de $LND_{EQ} - TRS_2$ ainda não foi feito. Felizmente, o presente trabalho está apenas interessados nas regras de $LND_{EQ} - TRS_2$ geradas pelo fato de que a *rw*-igualdade ser transitiva, reflexiva e simétrica, com adição de apenas uma regra específica do sistema $LND_{EQ} - TRS_2$. Ora, se a transtividade, reflexividade e simetria da igualdade geram regras como tt, tlr, sr, etc., então regras equivalentes deveriam ser geradas a partir da transitividade, reflexividade e simetria da *rw*-igualdade. Essas regras são facilmente obtidas. Para isso, é necessário colocar um nome nas sequências da *rw*-igualdade. Por exemplo, se s e t são *rw*-iguais devido a existência de uma sequência $\theta : R_0, ..., R_n$ tal que $R_0 = s$ e $R_n = t$, então pode-se dizer que $s =_{rw_\theta} t$. Com isso, obtém-se as seguintes regras:

- Regras envolvendo σ e ρ

$$\frac{x =_{rw_\rho} x : A}{x =_{rw_{\sigma(\rho)}} x : A} \quad \rhd_{sr_2} \quad x =_{rw_\rho} x : A$$

$$\frac{\dfrac{x =_{rw_r} y : A}{y =_{rw_{\sigma(r)}} x : A}}{x =_{rw_{\sigma(\sigma(r))}} y : A} \quad \rhd_{ss_2} \quad x =_{rw_r} y : A$$

- Regras envolvendo τ

$$\frac{x =_{rw_r} y : A \qquad y =_{rw_{\sigma(r)}} x : A}{x =_{rw_{\tau(r,\sigma(r))}} x : A} \quad \rhd_{tr_2} \quad x =_{rw_\rho} x : A$$

$$\frac{y =_{rw_{\sigma(r)}} x : A \qquad x =_{rw_r} y : A}{y =_{rw_{\tau(\sigma(r),r)}} y : A} \quad \rhd_{tsr_2} \quad y =_{rw_\rho} y : A$$

$$\frac{x =_{rw_r} y : A \qquad y =_{rw_\rho} y : A}{x =_{rw_{\tau(r,\rho)}} y : A} \quad \rhd_{trr_2} \quad x =_{rw_r} y : A$$

$$\frac{x =_{rw_\rho} x : A \qquad x =_{rw_r} y : A}{x =_{rw_{\tau(\rho,r)}} y : A} \quad \rhd_{tlr_2} \quad x =_{rw_r} y : A$$

- Regras envolvendoτ and τ

$$\frac{\dfrac{x =_{rw_t} y : A \qquad y =_{rw_r} w : A}{x =_{rw_{\tau(t,r)}} w : A} \qquad w =_{rw_s} z : A}{x =_{rw_{\tau(\tau(t,r),s)}} z : A}$$

$$\rhd_{tt_2} \frac{x =_{rw_t} y : A \qquad \dfrac{y =_{rw_r} w : A \qquad w =_{rw_s} z : A}{y =_{rw_{\tau(r,s)}} z : A}}{x =_{rw_{\tau(t,\tau(r,s))}} z : A}$$

Com isso, sao obtidas as regras que resolvem as redundâncias causadas pela transitividade, reflexividade e simetria da *rw*-igualdade. Além disso, como essas regras acima resolvem as redundancias da *rw*-igualdade, é dito que as regras são regras rw_2. Além disso, análogo a definição de *rw*-igualdade, é possível pensar em rw_2-igualdade, com a única diferença que as regras envolvidas são as rw_2 ao invés das *rw*.

A regra específica ao sistema $LND_{EQ}-TRS_2$ que é importante para o presente trabalho é originada a partir do fato que a transitividade de caminhos redutíveis pode ser reduzida a partir de formas diferentes, mas gerando o mesmo resultado. Por exemplo, considere o seguinte caso simples: a transitividade $\tau(s,t)$, com s redutível a s' e t redutível a t'. É possível reduzir $\tau(s,t)$ a partir de duas formas: A primeira é $\theta : \tau(s,t) \rhd_{1rw} \tau(s',t) \rhd_{1rw} \tau(s',t')$ e a segunda $\theta' : \tau(s,t) \rhd_{1rw} \tau(s,t') \rhd_{1rw} \tau(s',t')$. As duas sequências *rw* obtiveram o mesmo resultado de forma similar, apenas trocando a ordem em que as reduções foram feitas. Além da ordem das reduções, mais nada foi trocado. A redução de cada variável individual continua exatamente a mesma. Dessa forma, as duas sequências *rw* podem ser vistas como formas redundantes, uma em relação à outra. Portanto, pode-se estabelecer uma regra que resolve especificamente esses casos. A regra se chamará *indepedência da escolha* e será denotada por cd_2. Essa nova regra pode ser usada em casos bem mais complicados que o do exemplo visto. De fato, independente da quantidade de transitividades e variáveis, se a única diferença entre os caminhos de caminhos for as escolhas que foram feitas em cada etapa da transitividade, então essa regra é capaz de estabelecer a rw_2-igualdade entre as sequências *rw*. Essa lei será usada em um caso mais complexo para provar a lei da troca equivalente de uma bicategoria induzida por caminhos computacionais.

Proposição 5.1. *rw_2-igualdade é transitiva, simétrica e reflexiva.*

Demonstração: Análogo a prova da *proposição 4.1*.

□

Analogamente ao que foi feito para a *rw*-igualdade, é possível pensar que a rw_2-igualdade também gera redundâncias. Ora, já que a rw_2-igualdade é transitiva, simétrica e reflexiva, ao menos as redundâncias causadas por essas propriedades serão geradas. Dessa forma, pode-se pensar em regras rw_3 que resolvem essas redundâncias. Especificamente, pode-se pensar em tsr_3, tr_3, trr_3, etc. De fato, pode-se ir mais além:

Definição 5.1 (rw_n-igualdade)**.** *Uma* regra rw_n *resolve uma redundância causada pela* rw_{n-1}*-igualdade (ou apenas igualdade, para* $n = 1$*).*

O conjunto de regras rw_n formam o sistema $LND_{EQ} - TRS_n$. Além disso, análogo a *proposição 5.1*, rw_n-igualdade é transitiva, simétrica e reflexiva. Com isso, conclui-se a existência de infinitas *rw*-igualdades e sistemas $LND_{EQ} - TRS$.

5.2 Conceitos Básicos de Categorias de Alta Ordem

Os conceitos necessários da teoria das categorias de alta ordem serão mostrados nessa seção. As categorias serão construídas a partir dos diversos níveis de *rw*-igualdade que foram expostos na seção anterior.

5.2.1 Conjuntos Globulares

O primeiro conceito importante é o de conjuntos globulares:

Definição 5.2 (Conjuntos Globulares (LEINSTER, 2004))**.** *Seja* $n \in \mathbb{N}$*. Um conjunto globular* X *é o seguinte diagrama de conjuntos e funções:*

$$X(n) \underset{t}{\overset{s}{\rightrightarrows}} X(n-1) \underset{t}{\overset{s}{\rightrightarrows}} \cdots \underset{t}{\overset{s}{\rightrightarrows}} X(0)$$

O diagrama tem que obedecer as seguintes funções para todo $x \in \{2, ..., n\}$ *and* $x \in X(m)$*:*

$$s(s(x)) = s(t(x)), \quad t(s(x)) = t(t(x))$$

Os elementos $x \in X(n)$ *são conhecidos como n-células.*

Proposição 5.2. *Caminhos computacionais formam um* ∞*-conjunto-globular.*

Demonstração: Para ver que os caminhos computacionais formam um conjunto globular, é necessário entender o significado de uma n-célula. Dado um tipo A, pode-se considerar que uma 0-célula é simplesmente um termo $a : A$. Já uma 1-célula é um caminho entre duas 0-células, ou seja, um caminho computacional entre dois termos do tipo A. Uma 2-célula seria, então, uma rw-igualdade entre duas 1-células. Seguinte o mesmo raciocínio, uma n-célula é uma rw_{n-1}-igualdade entre duas $(n-1)$-células. Ora, dado n-célula qualquer θ e duas $n-1$-células x e y, terá-se que $x =_{rw_{(n-1)\theta}} y$. Define-se, então, que $s(\theta) = x$ e $t(\theta) = y$. Basta checar agora as equações globulares. Para isso, considere θ e α como n-células, com $x =_{rw_{(n-1)\theta}} y$ e $x =_{rw_{(n-1)\alpha}} y$. Agora considere uma $(n+1)$-célula ϕ tal que $\theta =_{rw_n\phi} \alpha$. Então, tem-se que:

$$s(s(\phi)) = s(\theta) = x = s(\alpha) = s(t(\phi))$$
$$t(s(\phi)) = t(\theta) = y = t(\alpha) = t(t(\phi)).$$

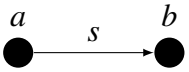
Portanto, as equações globulares se sustentam. Como são infinitas rw-igualdades, tem-se infinitas n-células e, portanto, os caminhos formam um ∞-conjunto-globular.

□

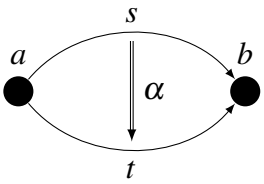
5.2.2 Composição Horizontal

Dado um conjunto globular, uma 0-célula é simplesmente um objeto e é representada pelo seguinte diagrama:

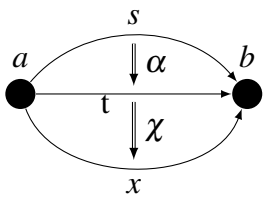
Já uma 1-célula é um morfismo entre objetos, sendo representada por:



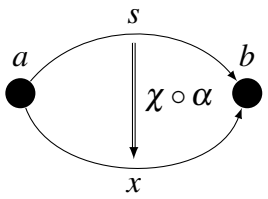
2-células são morfismos entre 1-células. Portanto, são representadas pelo seguinte diagrama:



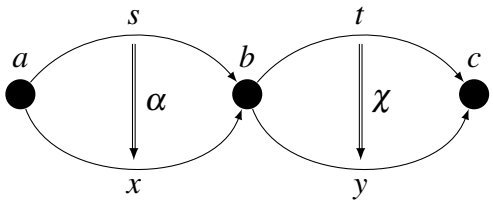
Como uma 2-célula é uma dimensão maior que 1-células, é possível compor 2-células de duas formas diferentes. A primeira forma é a composição tradicional, denotada por $\circ$. O diagrama



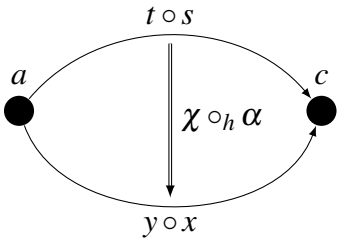
pode ser representado por:



Em um contexto de 2-células, a composição tradicional $\circ$ é também chamada de composição vertical. Para as 2-células, existe uma composição adicional que não está presente nas 1-células. Essa composição, denotada por $\circ_h$, é conhecida como composição horizontal (LEINSTER, 2004). É representada pela transformação do diagrama



no diagrama:



Portanto, ao trabalhar com categorias de ordem maior que 1, é necessário levar em conta esses novos tipos de composição. No caso das 2-categorias, as 2-células terão essas duas composições. Além disso, a composição horinzontal também terá que ser associativa e respeitar as leis de identidade. De forma análoga, as 3-células tem, além das duas composições das 2-células, uma composição adicional, já que é de uma dimensão maior. De fato, uma n-célula possuirá n composições. Formalmente, as m composições de m-células $\in A(m)$ são representadas por (LEINSTER, 2004):

$$A(m) \times_{A(p)} A(m) = \{(x', x) \in A(m) \times A(m) \mid t^{m-p}(x) = s^{m-p}(x')\}, \text{ com } p \leq m$$

5.2.3 Bicategorias

No capítulo passado, foi visto que os caminhos computacionais induzem uma categoria fraca. Nesse capítulo, o objetivo será mostrar que é possível induzir uma categoria fraca de alta ordem com duas dimensões. Essa categoria é chamada de bicategoria. Segue a definição:

Definição 5.3 (Bicategorias (LEINSTER, 1998))**.** *Uma* bicategoria Γ *consiste nos seguintes dados que obedecem os seguintes axiomas:*
Dados:

- *Coleção de objetos ob* Γ *(os elementos são* 0*-células* $A, B, \ldots$*)*
- *Categorias* $\Gamma(A,B)$ *(essas categorias tem objetos* 1*-células* $f, g, \ldots$ *e setas* 2*-células* $\alpha, \beta, \ldots$*)*
- *Funtores:*

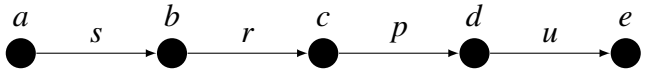
$$c_{ABC} : \Gamma(B,C) \times \Gamma(A,B) \to \Gamma(A,C)$$
$$(g, f) \to g \circ f = gf$$
$$(\beta, \alpha) \to \beta \circ_h \alpha$$

  e $1_A : 1 \to \Gamma(A,A)$ *(logo,* 1_A *é uma* 1*-célula* $A \to A$*)*

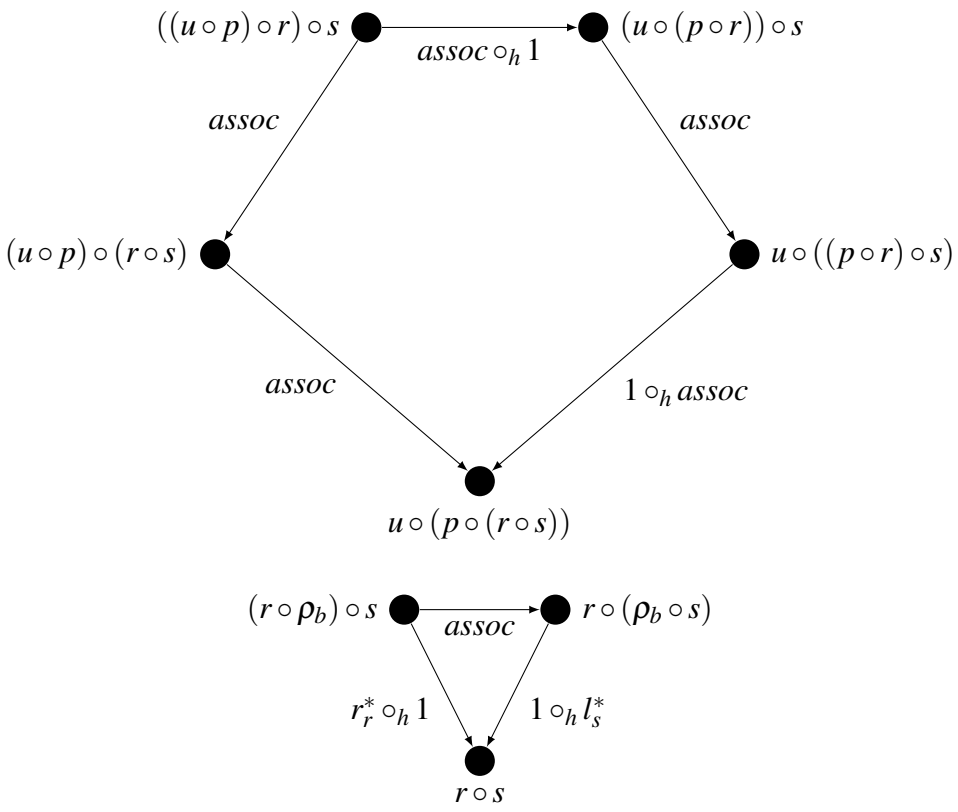
- *Os seguintes isomorfismos naturais:*

$$assoc_{hgf} : (hg)f \xrightarrow{\sim} h(gf)$$
$$r_f^* : f \circ 1_A \xrightarrow{\sim} f$$
$$l_f^* : 1_B \circ f \xrightarrow{\sim} f$$

Axiomas:
Dada a seguinte configuração de 1*-células*



os seguintes diagramas são comutativos:



Na literatura de categorias de alta ordem, os axiomas de uma categoria de alta ordem fraca são conhecidos como *leis de coerência* (LEINSTER, 2004).

5.3 Categorias de Alta Ordem Induzidas

O objetivo dessa seção é provar o importante resultado desse capítulo, isto é, o fato de que caminhos computacionais induzem uma bicategoria. Além disso, serão feitas conjecturas envolvendo categorias com mais que duas dimensões.

5.3.1 A Bicategoria Induzida

No capítulo passado, foi visto que um caminho computacional é capaz de induzir uma categoria fraca (que também é uma grupóide) chamada de A_{rw}. Como foi visto, os morfismos de A_{rw} são os caminhos entre os objetos do tipo A. Ora, o primeiro passo para se obter uma bicategoria é gerar uma categoria para cada par de objetos de A_{rw}. Dados $a, b : A$, pode-se pensar em $A_{2rw}(a,b)$ como sendo a seguinte estrutura:

Definição 5.4 ($A_{2rw}(a,b)$)**.** *$A_{2rw}(a,b)$ é uma estrutura com objetos e morfismos formados por:*

- *Objetos: caminhos computacionais s entre a e b, isto é, $a =_s b$.*
- *Morfismos: um morfismo é uma rw-igualdade entre dois caminhos computacionais, isto é, $\theta : s \to t$, com s e t caminhos e $s =_{rw_\theta} t$.*

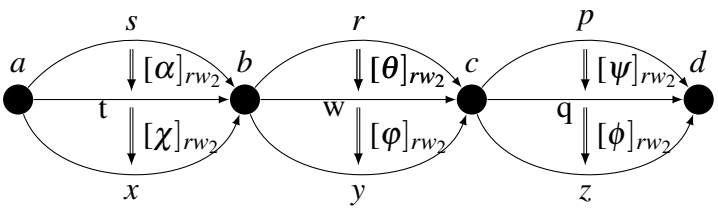
Dessa forma, pode-se pensar em uma estrutura $2-A_{rw}$, que possui a mesma estrutura de A_{rw} com a adição de estruturas A_{2rw} entre cada par de objetos de A_{rw}. À primeira vista, $2-A_{rw}$ parece ser uma possível bicategoria. O problema é que, como pode-se ver na *definição 5.3*, é necessário que A_{2rw} seja uma categoria no sentido estrito. Porém, as igualdades de A_{2rw} não se sustentam em um nível estrito. De forma análoga a A_{rw}, que apenas se sustenta a nível de rw-igualdade, A_{2rw} irá se sustentar apenas a nível de rw_2-igualdade. Para ver isso, basta olhar a prova da *proposição 4.2*. A única diferença será que regras rw_2 serão utilizadas no lugar das regras rw. Ou seja, serão utilizadas tt_2, tsr_2, etc. Portanto, A_{2rw} seria apenas uma categoria fraca. Felizmente, há uma forma simples de resolver esse problema. A solução é considerar uma estrutura $[A_{2rw}]$. Nessa estrutura, a única diferença é que os morfismos passariam a ser classes de equivalência da relação de rw_2-igualdade. Em outras palavras, se θ é um morfismo em A_{2rw}, então $[\theta]_{rw_2}$ passa a ser um morfismo de $[A_{2rw}]$. Além disso, a composição passa a ser definida como $[\theta]_{rw_2} \circ [\phi]_{rw_2} = [\theta \circ \phi]_{rw_2}$. Dessa forma, $[A_{2rw}]$ passaria a obedecer as regras de categoria de forma estrita. Com isso, pode-se pensar em A_{rw} com a adição de $[A_{2rw}](a,b)$ para cada par de objetos $a,b:A$. Essa estrutura será chamada de $[2-A_{rw}]$. Antes de provar que $[2-A_{rw}]$ é uma bicategoria, é necessário provar que todo morfismo de $[A_{2rw}]$ é um isomorfismo (ou seja, que $[A_{2rw}]$ é uma grupóide). Esse fato será utilizado posteriormente na prova que $[2-A_{rw}]$ é de fato uma bicategoria:

Proposição 5.3. *Todo morfismo de $[A_{2rw}]$ é um isomorfismo.*

Demonstração: Já que a rw_2-igualdade é simétrica, dado um $[\theta]_{rw_2}$, sempre é possível obter a inversa e $[\sigma(\theta)]_{rw_2}$. Então, tem-se $[\theta]_{rw_2} \circ [\sigma(\theta)]_{rw_2} = [\theta \circ \sigma(\theta)]_{rw_2}$. Porém, de $LND_{EQ}-TRS_2$ tem-se que $(\theta \circ \sigma(\theta)) = \tau(\sigma(\theta),\theta)) \rhd_{tsr_2} \rho_r$ e, portanto, $\theta \circ \sigma(\theta) =_{rw_2} \rho_r$. Logo, $[\theta \circ \sigma(\theta)]_{rw_2} = [\rho_r]_{rw_2}$. Analogamente, prova-se que $[\sigma(\theta) \circ \theta]_{rw_2} = [\rho_r]_{rw_2}$. Logo, qualquer morfismo de $[A_{2rw}]$ é um isomorfismo. □

Proposição 5.4. *A estrutura $[2-A_{rw}]$, induzida por caminhos computacionais, é uma bicategoria.*

Demonstração: Para provar essa proposição, o seguinte diagrama que representa $[2-A_{rw}]$ será utilizado:



Como foi visto anteriormente, uma bicategoria precisa ter uma composição horizontal. O primeiro passo da prova é definir essa composição. Dados $[\alpha]_{rw_2} : [s = \alpha_1, ..., \alpha_n = t]_{rw_2}$ e

$[\theta]_{rw_2} : [r = \theta_1, ..., \theta_m = w]_{rw_2}$, então define-se a composição horizontal $([\theta]_{rw_2} \circ_h [\alpha]_{rw_2})$ como a sequência $[\tau(s = \alpha_1, r = \theta_1), ..., \tau(\alpha_n, \theta_1)..., \tau(\alpha_n = t, \theta_m = w)]_{rw_2}$.

O proximo passo é definir os isomorfismos naturais. Para a associatividade *assoc*, é necessário checar a existência de um isomorfismo natural entre $(([\psi]_{rw_2} \circ_h [\theta]_{rw_2}) \circ_h [\alpha]_{rw_2})$ e $([\psi]_{rw_2} \circ_h ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}))$. Para isso, basta usar o fato provado na *proposição 3.7*, que indica que uma transformação natural é um isomorfismo sse todas as componentes são um isomorfismo. Dessa forma, basta provar que existe um isomorfismo para qualquer componente. Como se sabe que todo 2-morfismo de $[2 - A_{rw}]$ é um isomorfismo, então basta mostrar que existe um morfismo entre as componentes. Uma componente de $(([\psi]_{rw_2} \circ_h [\theta]_{rw_2})$ é um termo da forma $\tau(\alpha_x, \tau(\theta_y, \psi_z))$, com x, y e z naturais que obedecem a ordem estabelecida pela composição horizontal. De forma análoga, a componente equivalente de $([\psi]_{rw_2} \circ_h ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}))$ é o termo $\tau(\tau(\alpha_x, \theta_y), \psi_z)$ O morfismo entre esses termos é claramente estabelecido pela inversa da regra tt, ou seja, $\tau(\alpha_x, \tau(\theta_y, \psi_z)) =_{rw_{\sigma(tt)}} \tau(\tau(\alpha_x, \theta_y), \psi_z)$.

As leis de identidade usam a mesma ideia. Para l^*_s, é necessário checar que $([\alpha]_{rw_2} \circ_h [\rho_{\rho_a}]_{rw_2}) = [\alpha]_{rw_2}$. Para fazer isso, basca pegar componentes e provar o morfismo. Para $([\alpha]_{rw_2} \circ_h [\rho_{\rho_a}]_{rw_2})$, uma componente genérica é da forma $(\rho_{\rho_a}, \alpha_y)$ e para $[\alpha]_{rw_2}$, da forma α_y. O morfismo é estabelecido diretamente: $(\rho_{\rho_a}, \alpha_y) =_{rw_{tlr}} \alpha_y$.

Analogamente. r^*_s é estabelecido a partir de aplicações de trr.

Agora, é necessário estabelecer a lei da troca equivalente entre a composição normal e a horizontal, ou seja:

$$([\varphi]_{rw_2} \circ [\theta]_{rw_2}) \circ_h ([\chi]_{rw_2} \circ [\alpha]_{rw_2}) = ([\varphi]_{rw_2} \circ_h [\chi]_{rw_2}) \circ ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2})$$

Começando por $(([\varphi]_{rw_2} \circ [\theta]_{rw_2}) \circ_h ([\chi]_{rw_2} \circ [\alpha]_{rw_2}))$, tem-se:

$$\begin{gathered}(([\varphi]_{rw_2} \circ [\theta]_{rw_2}) \circ_h ([\chi]_{rw_2} \circ [\alpha]_{rw_2})) = \\ [\tau(\theta, \varphi)]_{rw_2} \circ_h [\tau(\alpha, \chi)]_{rw_2} = \\ [\theta_1, ..., \theta_n = \varphi_1, ..., \varphi_{n'}]_{rw_2} \circ_h [(\alpha_1, ..., \alpha_m = \chi_1, ...\chi_{m'}]_{rw_2} = \\ [\tau(\alpha_1, \theta_1), ..., \tau(\alpha_m = \chi_1, \theta_1), ..., \tau(\chi_n, \theta_1), ..., \tau(\chi_n, \theta_{m'} = \varphi_1), ..., \tau(\chi_n, \varphi_{n'})]_{rw_2}\end{gathered}$$

De $(([\varphi]_{rw_2} \circ_h [\chi]_{rw_2}) \circ ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2})))$:

$$\begin{gathered}(([\varphi]_{rw_2} \circ_h [\chi]_{rw_2}) \circ ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2})) = \\ ([\tau(\chi_1, \varphi_1), ..., \tau(\chi_n, \varphi_1), ..., \tau(\chi_n, \varphi_{n'})]_{rw_2} \circ [\tau(\alpha_1, \theta_1), ..., \tau(\alpha_m, \theta_1), ..., \tau(\alpha_m, \theta_{m'})]_{rw_2} = \\ [\tau(\alpha_1, \theta_1), ..., \tau(\alpha_m, \theta_1), ..., \tau(\alpha_m, \theta_{m'}), ..., \tau(\chi_1, \varphi_1), ..., \tau(\chi_n, \varphi_1), ..., \tau(\chi_n, \varphi_{n'})]_{rw_2}\end{gathered}$$

A princípio, os dois resultados parecem diferentes. Porém, ao olhar atentamente, percebe-se que esse é um caso perfeito para o uso da regra da escolha independente. Ora, pode-se ver que as expansões individuais de cada varíavel são exatamente iguais, apenas foi modificada a escolha de cada passo da transitividade. Portanto, tem-se que $[\tau(\alpha_1, \theta_1), ..., \tau(\alpha_m = \chi_1, \theta_1), ..., \tau(\chi_n, \theta_1), ..., \tau(\chi_n, \theta_{m'} = \varphi_1), ..., \tau(\chi_n, \varphi_{n'})]_{rw_2} =_{cd_2} [\tau(\alpha_1, \theta_1), ..., \tau(\alpha_m, \theta_1), ..., \tau(\alpha_m, \theta_{m'})$,

$..., \tau(\chi_1, \varphi_1), ..., \tau(\chi_n, \varphi_1), ..., \tau(\chi_n, \varphi_{n'})]_{rw_2}$. Já que está se trabalhando com classes de equivalência, a igualdade se sustenta de forma estrita.

Por fim, é necessário checar as leis de coerência. Olhando para a *definição 5.3*, é necessário provar a comutatividade do pentágono e do triângulo. Para o pentágono, pode-se começar com $((u \circ p) \circ r) \circ s = \tau(s, \tau(r, \tau(p, u)))$ e ir para a direita:

$$(assoc \circ_h [\rho_s]_{rw_2})(\tau(s, \tau(r, \tau(p, u)))) = \tau(s, assoc(\tau(r, \tau(p, u)))) =$$
$$\tau(s, \tau(\tau(r, p), u))$$
$$assoc(\tau(s, \tau(\tau(r, p), u))) = \tau(\tau(s, \tau(r, p)), u)$$
$$([\rho_u]_{rw_2} \circ_h assoc)(\tau(\tau(s, \tau(r, p)), u)) = \tau(assoc(\tau(s, \tau(r, p))), u) =$$
$$\tau(\tau(\tau(s, r), p)), u) = u \circ (p \circ (r \circ s))$$

Começando agora pelo mesmo $((u \circ p) \circ r) \circ s = \tau(s, \tau(r, \tau(p, u)))$ e indo para baixo-esquerda:

$$assoc(\tau(s, \tau(r, \tau(p, u)))) = \tau(\tau(s, r), \tau(p, u))$$
$$assoc(\tau(\tau(s, r), \tau(p, u))) = \tau(\tau(\tau(s, r), p), u) = u \circ (p \circ (r \circ s))$$

Logo, o diagrama comuta, já que foram obtidos os mesmos resultados a partir de caminhos diferentes. Para o triângulo, pode-se começar por $((r \circ \rho_b) \circ s) = \tau(s, \tau(\rho_b, r))$ e ir para a direita:

$$assoc(\tau(s, \tau(\rho_b, r))) = \tau(\tau(s, \rho_b), r)$$
$$([\rho_r]_{rw_2} \circ_h l_s^*)\tau(\tau(s, \rho_b), r) = \tau(l_s^*(\tau(s, \rho_b)), r) =$$
$$\tau(s, r) = r \circ s$$

Agora, começando pelo mesmo $((r \circ \rho_b) \circ s) = \tau(s, \tau(\rho_b, r))$ e indo para baixo-direita:

$$(r_r^* \circ_h [\rho_s]_{rw_2})\tau(s, \tau(\rho_b, r)) = \tau(s, r_r^*(\tau(\rho_b, r))) =$$
$$\tau(s, r) = r \circ s$$

Portanto, o segundo diagrama comuta e a prova está finalizada. □

Proposição 5.5. $[2 - A_{rw}]$ *é uma* 2*-grupóide fraca.*

Demonstração: Conseqüencia direta do fato que $[2 - A_{rw}]$ é uma bicategoria (*proposição 5.4*), A_{rw} é uma grupóide fraca (*proposição 4.3*) e $[A_{2rw}]$ uma grupóide (*proposição 5.3*). □

5.3.2 Categorias Com Mais de Duas Dimensões

Na subseção anteiror, foi construída uma prova formal do fato que os caminhos computacionais induzem uma bicategoria. Da mesma forma que existem as categorias com duas dimensões, é possível pensar em categorias com três ou mais dimensões. Na verdade, é até mesmo razoável pensar em uma categoria com infinitas dimensões, chamada de ω-categoria. O grande problema de se trabalhar com categorias de alta ordem, é a complexidade inerente dessas entidades. Como foi visto, o próprio conceito de bicategoria já envolve propriedades complexas e que exigem um certo trabalho para serem verificadas. No caso de categorias de níveis maiores, é necessário checar mais composições (já que uma n-célula possui n composições) além de que os axiomas (ou seja, as leis de coerência) tornam-se cada vez mais complexos. Portanto, provar que os caminhos computacionais induzem categorias com ordem maior que 2 é um trabalho não-trivial. De fato, esse trabalho se limitará a provar apenas o que já foi feito na seção passada. Porém, é possível pensar em conjecturas, as quais poderão se tornar teoremas a partir de trabalhos futuros. A ideia é que, já que os caminhos induziram A_{rw}, a qual é uma categoria fraca e $[2-A_{rw}]$, que é uma bicategoria, a ideia é que a adição de uma nova camada irá resultar em uma 3-categoria fraca (também chamada de tricategoria). Ora, para isso, seria necessário usar $2-A_{rw}$ e adicionar para cada par de 2-morfismos uma nova categoria A_{3rw}, em que os morfismos seriam rw_2-igualdades módulo rw_3-igualdade. Nesse caso, seria gerada uma estrutura $[3-A_{rw}]$. O desafio seria, então, provar que essa estrutura é realmente uma tricategoria. Além disso, usando o mesmo raciocínio utilizado para $[2-A_{rw}]$, $[3-A_{rw}]$ passsaria a ser uma 3-grupóide fraca. Estendendo essa construção para um número arbitrário de camadas, seria possível pensar em $[n-A_{rw}]$. Então, a conjectura seria a seguinte:

Conjectura 5.1. *$[n-A_{rw}]$ é uma n-grupóide fraca.*

Esse possível resultado pode ser ainda expandido. Ao invés de limitar a construção para um número n de camadas, pode-se pensar em uma estrutura com infinitas dimensões, já que existem infinitas rw-igualdades. Portanto, caminhos computacionais induzem uma estrutura fraca com infinitos níveis denotada por $\omega-A_{rw}$.

Conjectura 5.2. *$[\omega-A_{rw}]$ é uma ω-grupóide fraca.*

Transformar essas conjecturas em teoremas será resultado de grande importância e será o foco de trabalhos futuros. A possibilidade dessas conjecturas serem realmente teoremas é alta, já que foi provado por LUMSDAINE (2009); BERG; GARNER (2011) que o tipo identidade tal qual originalmente formulado por Martin-Löf realmente induz uma ω-grupóide fraca.

5.4 Considerações Finais Sobre Categorias de Alta Ordem

Esse capítulo serviu para complementar os resultados iniciados no capítulo anterior. O conceito de categoria foi extendido para uma versão mais completa conhecida como categorias

de alta ordem. A possibilidade de caminhos computacionais induzirem categorias de alta ordem surgiu após a descoberta de infinitas *rw*-igualdades. Utilizando os diversos níveis de *rw*-igualdades juntamente com conceitos básicos de teoria de alta ordem, foi possível provar que os caminhos computacionais induzem uma estrutura de dois níveis conhecida como bicategoria. A partir do processo utilizado para obter essa estrutura, foi mostrado que é possível expandi-lo, obtendo-se estruturas de níveis ainda maiores. Devido à dificuldade inerente de se trabalhar com essas estruturas que possuem diversos níveis, conjecturas foram feitas acerca de propriedades que essas estruturas possivelmente revelarão. O objetivo de transformar essas conjecturas em possíveis proposições e teoremas foi postergado para trabalhos futuros. Esse capítulo também marca os últimos resultados matemáticos envolvendo caminhos computacionais que foram obtidos no presente trabalho.

6

Conclusão

Inspirado pela existência de uma entidade computacional que estabelece a igualdade entre dois objetos computacionais, conhecida como caminhos computacionais, esse trabalho teve dois grandes objetivos. O primeiro foi mostrar que essa identidade pode ser entendida como um conceito formal da teoria dos tipos. Especificamente, sao os termos que possuem o tipo identidade como tipo. O segundo objetivo foi estudar as propriedades matemáticas dessa entidade. Para isso, foram utilizados os conceitos e a semântica da teoria das categorias. O primeiro objetivo foi desenvolvido no *capítulo 2*, enquanto o segundo foi desenvolvido nos *capítulos 3,4,5*.

A introdução do trabalho foi dedicada a fazer uma breve introdução sobre os fundamentos da matemática e explicar os problemas associados à teoria que é mais utilizada para servir como fundamento, que é a teoria axiomática dos conjuntos conhecida como ZFC. Além disso, a introdução explicou como o trabalho seria organizado e dividido.

O *capítulo 2* focou inicialmente em introduzir adequadamente a teoria dos tipos. Para isso, foi mostrado diversos conceitos básicos e essenciais, como o conceito de função dependente. Também foi mostrado como é possível construir os naturais a partir dessa teoria. Após os conceitos basicos serem devidamente introduzidos, um dos conceitos mais essenciais foi explicado, que é o tipo identidade. Para isso, foi mostrada a abordagem atual com detalhes. Além disso, foi explicado que a abordagem atual é demasiamente complexa e não intuitiva. O próximo passo foi finalmente introduzir a entidade conhecida como caminhos computacionais. Com isso, foi possível mostrar que os caminhos computacionais podem ser entendidos como os termos de um tipo identidade. A partir dessa interpretação, foi possível introduzir novas regras para o tipo identidade. Regras essas que mostraram-se muito mais intuitivas e facéis de serem utilizadas em comparação com o tipo identidade tradicional.

O *capítulo 3* foi inteiramente dedicado à introdução dos principais conceitos da teoria das categorias. Foi feito uma explicação detalhada da definição de categoria, além de ser mostrados conceitos básicos como dualidade, comutatividade, produtos, isomorfismos, etc. Posteriormente, conceitos mais avançados, como o de transformação natural, foram introduzidos. A ideia desse capítulo foi montar a base necessária para obter os resultados dos capítulos posteriores, além de

mostrar uma relação entre teoria dos tipos e teoria das categorias.

O *capítulo 4* é o primeiro capítulo responsável por mostrar resultados matemáticos envolvendo os caminhos computacionais. Para isso, foram utilizados os conceitos básicos definidos no *capítulo 3*. Inicialmente, foram mostrados exemplos que deixam claro a possibilidade de existir redundâncias dentro de um caminho computacional. Após isso, foi mostrado a existência de um sistema que mapeia e resolve essas redundâncias. A partir das regras desse sistema, foi possível, a partir do conceito de caminho computacional, induzir uma estrutura que possui as propriedades de categoria. Especificamente, foi feito uma diferenciação entre uma categoria fraca e uma estrita e, a partir dessas diferenças, mostrar que a estrutura induzida obdece as regras de uma categoria fraca. O resultado foi estendido, provando que a estrutura também é uma grupóide fraca. Com isso, os conceitos de categorias permitiram interpretar algumas propriedades matemáticas dos caminhos computacionais. Além disso, foi dito que essas propriedades estão de acordo com resultados anteriormente obtidos por HOFMANN; STREICHER (1994) envolvendo o tipo identidade tradicional.

Por fim, o objetivo do *capítulo 5* foi expandir os resultados obtidos no *capítulo 4*. Para isso, foi utilizada uma forma mais avançada de categoria, conhecida como categoria de alta ordem. O capítulo começa introduzindo a existência de diversos níveis de *rw*-igualdade. Logo depois, introduz os conceitos básicos de cateogiras de alta ordem. A partir desses conceitos e dos diversos niveis de *rw*-igualdade, é provado que é possível induzir estruturas de 2 níveis conhecidas como bicategorias. Por fim, são feitas conjecturas envolvendo estruturas de diversos níveis.

6.1 Trabalhos Futuros

Diversas áreas desse trabalho podem ser expandidas em possíveis trabalhos futuros. Uma primeira ideia é expandir o estudo da relação entre caminhos computacionais e o tipo identidade. Uma grande possibilidade de trabalho nessa área é provar diversos teoremas da teoria homotópica dos tipos utilizando a abordagem de caminhos ao invés da abordagem tradicional do tipo identidade. Para isso, o alvo inicial seriam os diversos teoremas presentes em UNIVALENT FOUNDATIONS PROGRAM (2013). Uma segunda possibilidade seria explorar os conceitos da teoria das categorias e mostrar como esses conceitos poderiam ser possivelmente expressos a partir de caminhos computacionais. Seria uma área promissora, já que a teoria dos tipos e a teoria das categorias estão diretamente conectadas. Entretanto, os resultados que complementariam diretamente os resultados obtidos nesse trabalho seria um trabalho que focasse em provar as conjecturas propostas no capítulo 5. Isso tornaria as estruturas dos caminhos computacionais condizentes com os resultados obtidos por LUMSDAINE (2009); BERG; GARNER (2011), os quais provam que o tipo identidade tradicional induz uma ω-grupóide fraca.

Referências

ALAMA, J. The Lambda Calculus. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Spring 2015.ed. [S.l.: s.n.], 2015.

AVIGAD, J. Philosophy of mathematics. In: BOUNDAS, C. (Ed.). **The Edinburgh Companion to Twentieth-Century Philosophies**. [S.l.]: Edinburgh University Press, 2007. p.234–251.

AWODEY, S. **Category theory**. 2nd.ed. [S.l.]: Oxford University Press, 2010.

AWODEY, S. **Category Theory Foundations**. Category Theory Foundations, Aula ministrada no programa de verão em linguagens de programação da Universidade de Oregon, Eugene, Oregon.

BARKER-PLUMMER, D. Turing Machines. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Summer 2013.ed. [S.l.: s.n.], 2013.

BERG, B. van den; GARNER, R. Types are weak ω-groupoids. **Proceedings of the London Mathematical Society**, [S.l.], v.102, n.2, p.370–394, 2011.

BRIDGES, D.; PALMGREN, E. Constructive Mathematics. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2013.ed. [S.l.: s.n.], 2013.

DUMMETT, M. **Elements of Intuitionism**. [S.l.]: Oxford, 1977. (Oxford Logic Guides, v.39).

HALLETT, M. Zermelo's Axiomatization of Set Theory. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Fall 2013.ed. [S.l.: s.n.], 2013.

HARPER, R. **Type Theory Foundations**. Type Theory Foundations, Aula ministrada no programa de verão em linguagens de programação da Universidade de Oregon, Eugene, Oregon.

HINDLEY, J. R.; SELDIN, J. P. **Lambda-calculus and combinators**: an introduction. [S.l.]: Cambridge University Press, 2008.

HOFMANN, M.; STREICHER, T. The groupoid model refutes uniqueness of identity proofs. In: LOGIC IN COMPUTER SCIENCE, 1994. LICS'94. PROCEEDINGS., SYMPOSIUM ON. **Anais...** IEEE, 1994. p.208–212.

HORSTEN, L. Philosophy of Mathematics. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Spring 2015.ed. [S.l.: s.n.], 2015.

HOWARD, W. A. The formulas-as-types notion of construction. In: SELDIN, J. P.; HINDLEY, J. R. (Ed.). **To H. B. Curry**: Essays on Combinatory Logic, Lambda Calculus, and Formalism. [S.l.]: Academic Press, 1980. p.479–490. Reprint of 1969 article.

HRBACEK, K.; JECH, T. **Introduction to Set Theory, Revised and Expanded**. [S.l.]: Chapman & Hall/CRC, 1999. (Pure and Applied Mathematics, v.220).

IRVINE, A. D. Bertrand Russell. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2014.ed. [S.l.: s.n.], 2014.

LEINSTER, T. Basic bicategories. **arXiv preprint math.CT/9810017**, [S.l.], 1998.

LEINSTER, T. **Higher Operads, Higher Categories**. [S.l.]: Cambridge University Press, 2004. (London Mathematical Society Lecture Note Series (Book 298)). Also http://arxiv.org/abs/math/0305049, May, 2003.

LUMSDAINE, P. L. Weak ω-categories from intensional type theory. In: TYPED LAMBDA CALCULI AND APPLICATIONS. **Anais...** Springer, 2009. p.172–187. (LNCS, v.5608).

MARQUIS, J.-P. Category Theory. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2014.ed. [S.l.: s.n.], 2014.

MARTIN-LÖF, P. An Intuitionistic Theory of Types: predicative part. In: ROSE, H.; SHEPHERDSON, J. (Ed.). **Logic Colloquium '73 Proceedings of the Logic Colloquium**. [S.l.]: Elsevier, 1975. p.73 – 118. (Studies in Logic and the Foundations of Mathematics, v.80).

MARTIN-LÖF, P. Constructive mathematics and computer programming. In: COHEN, L. et al. (Ed.). **Logic, Methodology and Philosophy of Science VI, Hannover, 1979**. [S.l.]: North-Holland, 1982. p.153–175. (Studies in Logic and the Foundations of Mathematics, v.104).

MARTIN-Löf, P. **Intuitionistic Type Theory. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980**. [S.l.]: Bibliopolis, Napoli, 1984. xii+92ppp.

MARTIN-LÖF, P. An intuitionistic theory of types. In: SAMBIN, G.; SMITH, J. (Ed.). **Twenty-five years of constructive type theory**. [S.l.]: Oxford University Press, 1998. p.127–172. (Oxford Logic Guides, v.36).

NLAB. **product**. Http://ncatlab.org/nlab/show/product (Versão 19). Última revisão por Urs Schreiber em 19 de Julho de 2014. Acessado em 24 de março de 2015.

OLIVEIRA, A. G. de. **Proof transformations for labelled natural deduction via term rewriting**. Master's thesis, Depto. de Informática, Universidade Federal de Pernambuco, Recife, Brazil, April 1995.

OLIVEIRA, A. G. de; QUEIROZ, R. J. G. B. de. Term rewriting with labelled deductive systems. In: BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE (SBIA'94). **Proceedings...** [S.l.: s.n.], 1994. p.59–72.

OLIVEIRA, A. G. de; QUEIROZ, R. J. G. B. de. A normalization procedure for the equational fragment of labelled natural deduction. **Logic Journal of IGPL**, [S.l.], v.3, n.(2–3), p.243–290, 1999.

QUEIROZ, R. J. G. B. de; GABBAY, D. The functional interpretation of the existential quantifier. **Bulletin of the IGPL**, [S.l.], v.7, n.2, p.173–215, 1995. (Special Issue on Deduction and Language, Guest Editor: Ruth Kempson). Full version of a paper presented at Logic Colloquium '91, Uppsala. Abstract in JSL 58(2):753–754, 1993.

QUEIROZ, R. J. G. B. de; GABBAY, D. M. Equality in Labelled Deductive Systems and the functional interpretation of propositional equality. In: AMSTERDAM COLLOQUIUM, 9. **Proceedings...** [S.l.: s.n.], 1994. p.547–565.

QUEIROZ, R. J. G. B. de; OLIVEIRA, A. G. de. **Propositional equality, identity types, and direct computational paths**. http://arxiv.org/abs/1107.1901, 2011 (latest version 2013).

QUEIROZ, R. J. G. B. de; OLIVEIRA, A. G. de. **Propositional Equality, Identity Types and Reversible Rewriting Sequences as Homotopies**. Propositional Equality, Identity Types and Reversible Rewriting Sequences as Homotopies, Palestra ministrado no Workshop de Lógica, Universidade Federal do Ceará, Fortaleza, CE.

QUEIROZ, R. J. G. B. de; OLIVEIRA, A. G. de. Natural deduction for equality: The missing entity. In: PEREIRA, L.; HAEUSLER, E.; PAIVA, V. de (Ed.). **Advances in Natural Deduction**. [S.l.]: Springer, 2014. p.63–91.

QUEIROZ, R. J. G. B. de; OLIVEIRA, A. G. de; GABBAY, D. M. **The Functional Interpretation of Logical Deduction**. [S.l.]: World Scientific, 2011.

RAMOS, A. F.; QUEIROZ, R. J. G. B. de; OLIVEIRA, A. G. de. **On the Identity Type as the Type of Computational Paths**. http://arxiv.org/abs/1504.04759, 2015. Aceito para apresentação no *LSFA 2015 - 10th Workshop on Logical and Semantic Frameworks, with Applications*, Natal, RN, 31-Ago a 01-Set, 2015.

RAMOS, A. F.; QUEIROZ, R. J. G. B. de; OLIVEIRA, A. G. de. **On the Groupoid Model of Computational Paths**. http://arxiv.org/abs/1506.02721, 2015. Aceito para apresentação no *LSFA 2015 - 10th Workshop on Logical and Semantic Frameworks, with Applications*, Natal, RN, 31-Ago a 01-Set, 2015.

UNIVALENT FOUNDATIONS PROGRAM. **Homotopy Type Theory**: univalent foundations of mathematics. Institute for Advanced Study: `http://homotopytypetheory.org/book`, 2013.

VOEVODSKY, V. **Univalent Foundations and Set Theory**. Univalent Foundations and Set Theory, Aula no Instituto de Pesquisa Avançada, Princeton, New Jersey.