Relações Binárias

Roteiro

- **Relações Binárias**
  - Raciocinar sobre conexões entre objetos.
- **Relações de Equivalência**
  - Raciocinando sobre clusters.
- **Um Teorema Fundamental**
  - Como sabemos que temos a definição “certa” para algo?

Relacionamentos

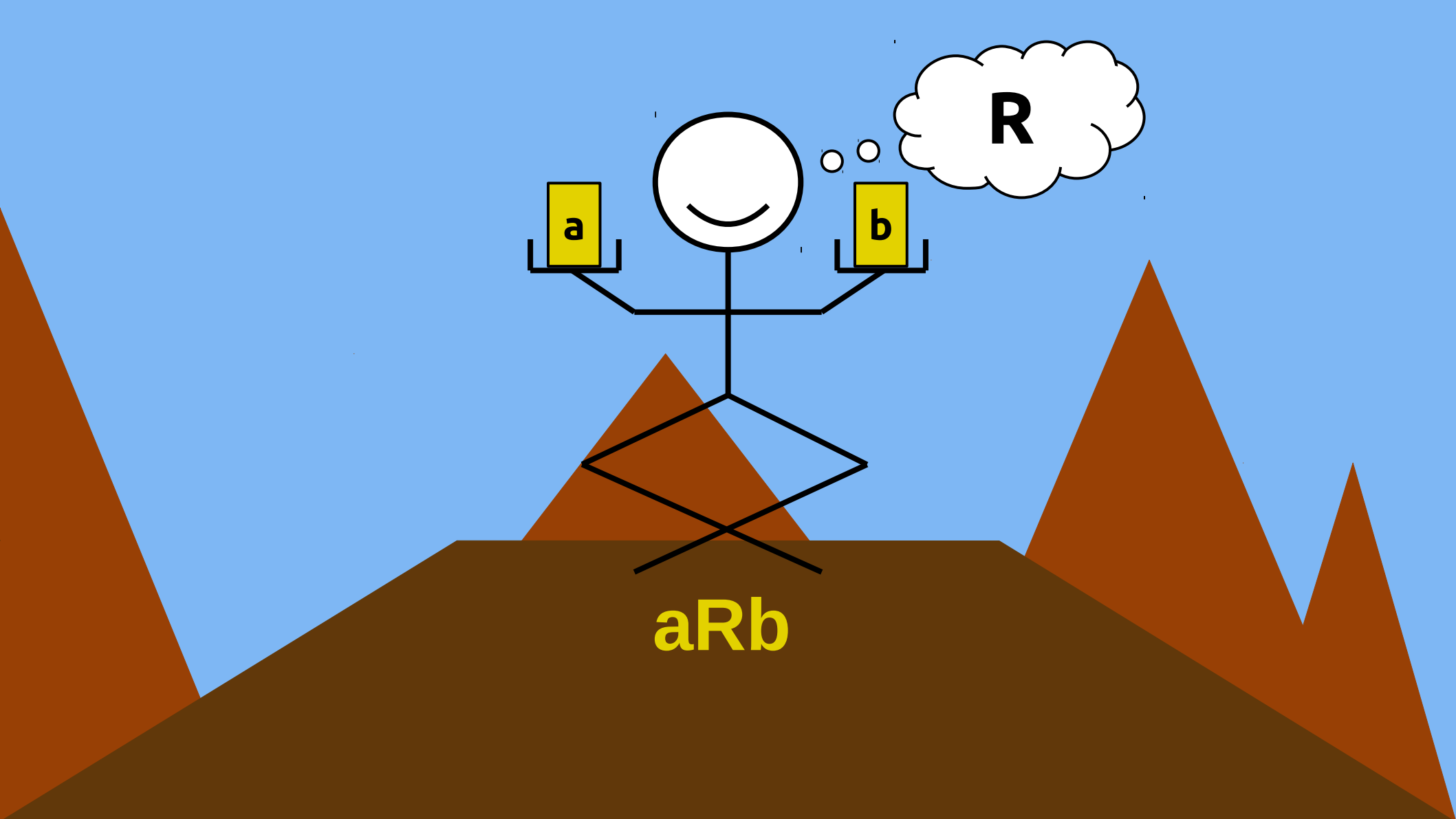
- Até então vimos exemplos de relacionamentos
  - entre conjuntos:

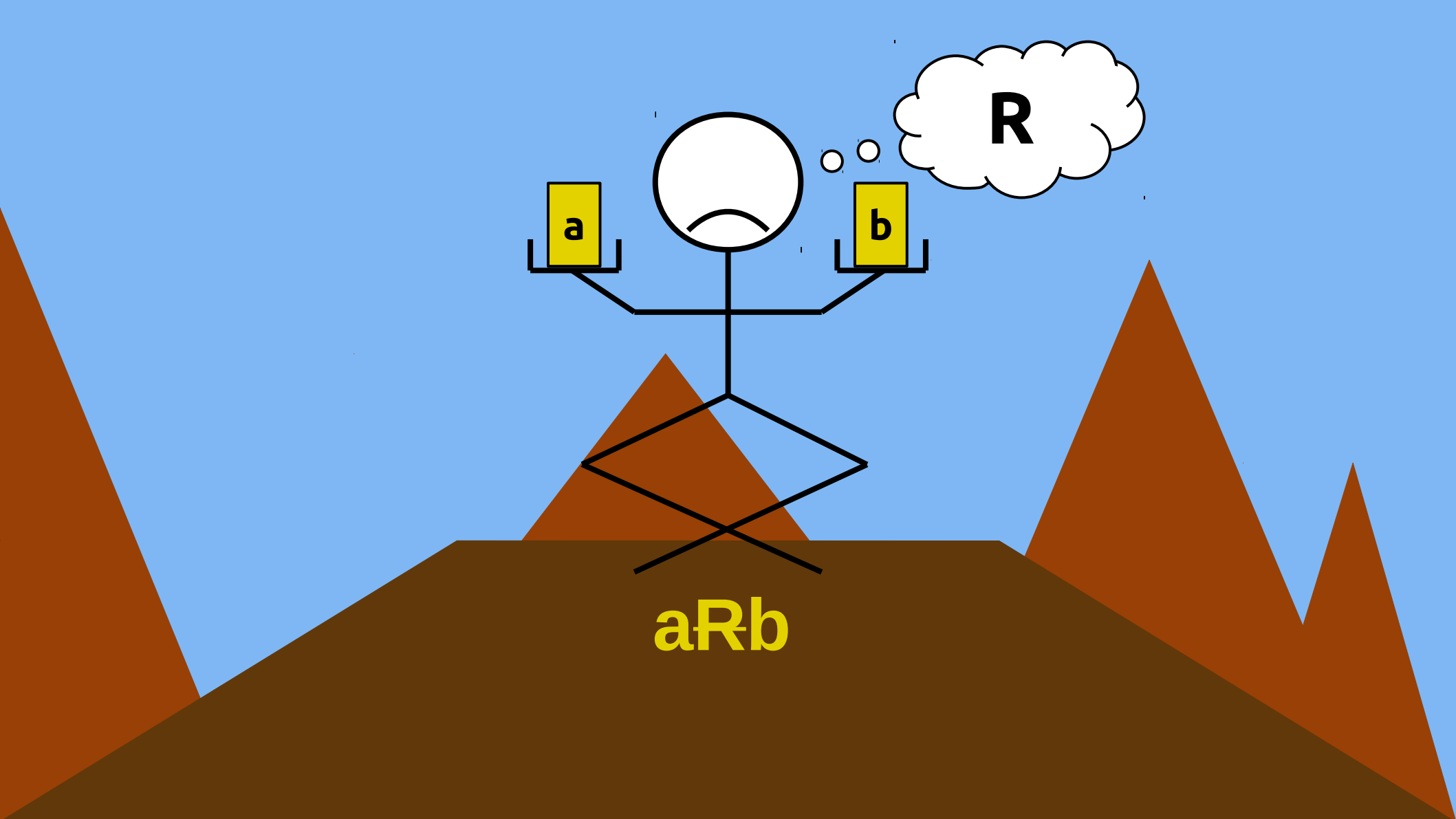
    $A \subseteq B$
  - entre os números:

    $x < y$ $\quad$ $x \equiv_k y$ $\quad$ $x \leq y$
  - entre pessoas:

    p ama q
- Como essas relações se concentram nas conexões entre dois objetos, são chamadas de **relações binárias**.
  - O "binário" aqui significa "pertencente a duas coisas", não "feito de zeros e uns".

O que exatamente é uma relação binária?


R
a
b
aRb


R
a
b
aRb

Relações Binárias

- Uma **relação binária sobre um conjunto A** é um predicado R que pode ser aplicado a pares de elementos extraídos de A.
- Se R é uma relação binária sobre A e vale para o par (a, b), escrevemos **aRb**.

$3 = 3$ $5 < 7$ $\varnothing \subseteq \mathbb{N}$

- Se R é uma relação binária sobre A e não vale para o par (a, b), escrevemos **a$\not$Rb**.

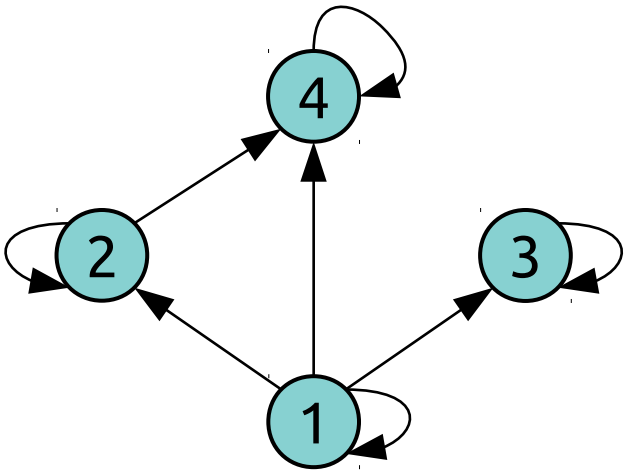
$4 \neq 3$ $4 \not< 3$ $\mathbb{N} \subseteq \varnothing$

Propriedades das Relações

- De modo geral, se R é uma relação binária sobre um conjunto A, a ordem dos operandos é significativa.
  - Por exemplo, $3 < 5$, mas $5 \not< 3$.
  - Em algumas relações, a ordem é irrelevante.
- As relações são sempre definidas em relação a algum conjunto subjacente.
  - Não faz sentido perguntar se ☺ $\subseteq$ 15, por exemplo, uma vez que $\subseteq$ é definida sobre conjuntos, não objetos arbitrários.

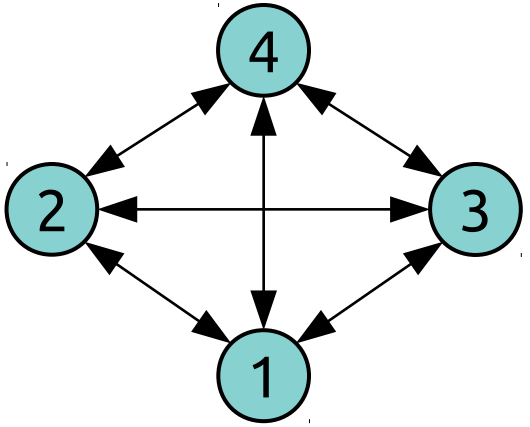
Visualizando Relações

- Podemos visualizar uma relação binária R sobre um conjunto A desenhando os elementos de A e desenhando uma linha entre um elemento a e um elemento b se aRb for verdadeiro.
- Exemplo: a relação a | b (que significa “a divide b”) sobre o conjunto {1, 2, 3, 4} tem a seguinte aparência:



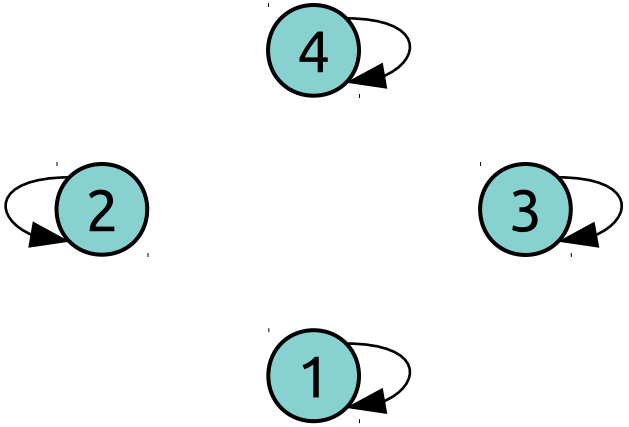
Visualizando Relações

- Podemos visualizar uma relação binária R sobre um conjunto A desenhando os elementos de A e desenhando uma linha entre um elemento a e um elemento b se aRb for verdadeiro.
- Exemplo: a relação a ≠ b sobre o conjunto {1, 2, 3, 4} é assim:



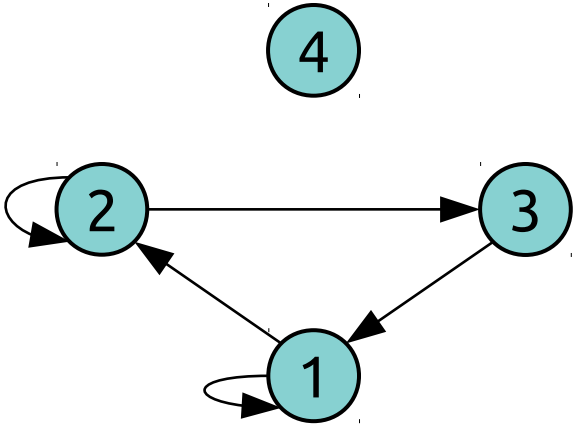
Visualizando Relações

- Podemos visualizar uma relação binária R sobre um conjunto A desenhando os elementos de A e desenhando uma linha entre um elemento a e um elemento b se aRb for verdadeiro.
- Exemplo: a relação a = b sobre o conjunto {1, 2, 3, 4} é assim:



Visualizando Relações

- Podemos visualizar uma relação binária R sobre um conjunto A desenhando os elementos de A e desenhando uma linha entre um elemento a e um elemento b se aRb for verdadeiro.
- Exemplo: abaixo está alguma relação sobre {1, 2, 3, 4} que é uma relação totalmente válida, embora não pareça haver uma regra unificadora simples.

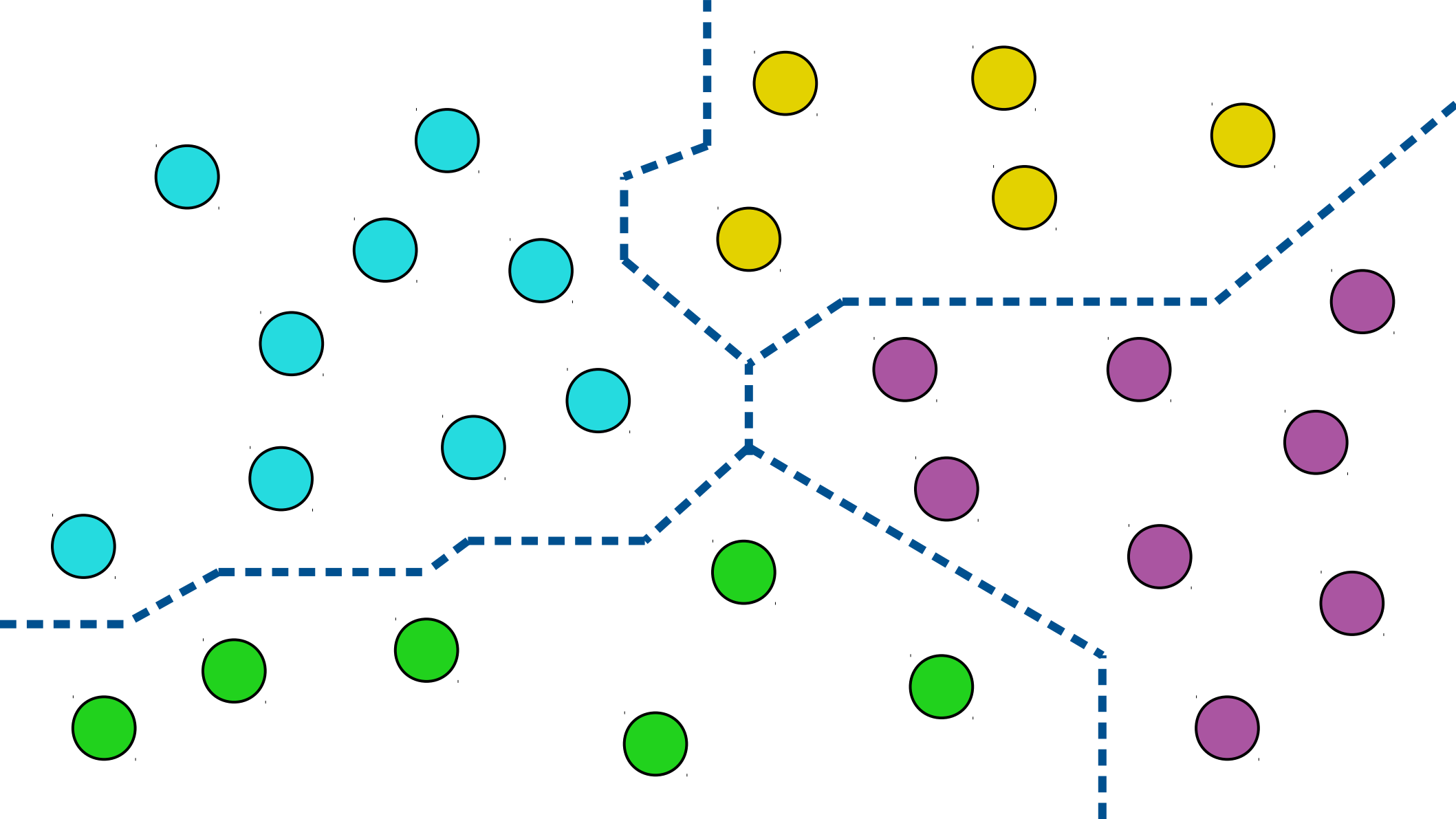


Capturando Estrutura

Capturando Estrutura

- As relações binárias são uma excelente forma de capturar certas estruturas que aparecem na ciência da computação.
- Nesta apresentação, veremos uma delas (**partições**), e na próxima veremos outra (**pré-requisitos**).
- Ao longo do caminho, exploraremos como escrever provas sobre as definições fornecidas na lógica de primeira ordem.

Partições



Partições

- Uma **partição de um conjunto** é uma forma de dividir o conjunto em subconjuntos não vazios e separados, de forma que cada elemento pertença exatamente a um subconjunto.
  - Dois conjuntos são **disjuntos** se sua interseção for o conjunto vazio; formalmente, os conjuntos S e T são disjuntos se $S \cap T = \emptyset$.
- Intuitivamente, uma partição de um conjunto divide o conjunto em pedaços menores.
- Não precisa haver nenhuma rima ou razão para o que essas peças são, embora muitas vezes haja uma.

Partições e Clusterização

- Se você tem um conjunto de dados, geralmente pode aprender algo com os dados, encontrando uma partição “boa” desses dados e inspecionando as partições.
  - Normalmente, o termo **clusterização** é usado na análise de dados em vez de particionamento.

Qual é a conexão entre as partições e relações binárias?

$$\forall a \in A.\ aRa$$

$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow bRa)$$

$$\forall a \in A.\ \forall b \in A.\ \forall c \in A.(aRb \wedge bRc \rightarrow aRc)$$

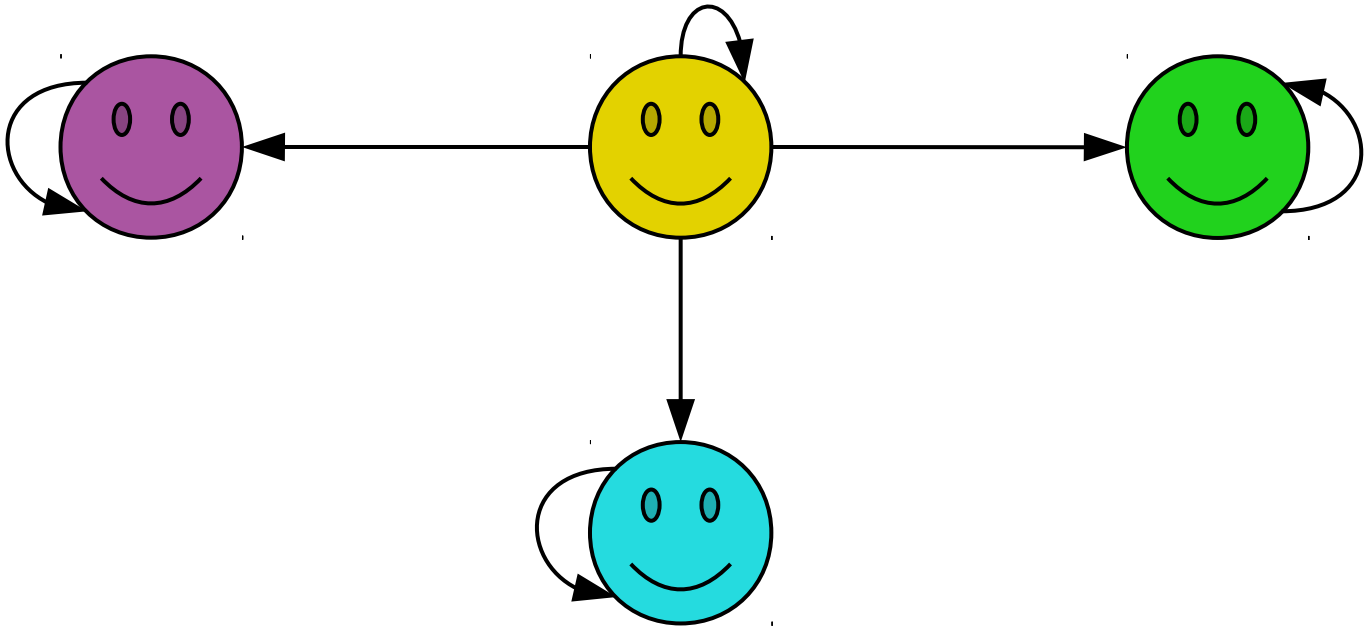
Reflexividade

- Algumas relações sempre se mantêm de qualquer elemento para si mesmo.
- Exemplos:
  - $x = x$ para qualquer x.
  - $A \subseteq A$ para qualquer conjunto A.
  - $x \equiv_k x$ para qualquer x.
- Relações desse tipo são chamadas de **reflexivas**.
- Falando formalmente, uma relação binária R sobre um conjunto A é reflexiva se a seguinte afirmação de primeira ordem é verdadeira:

  $\forall a \in A.\ aRa$

  ("Cada elemento está relacionado consigo mesmo.")

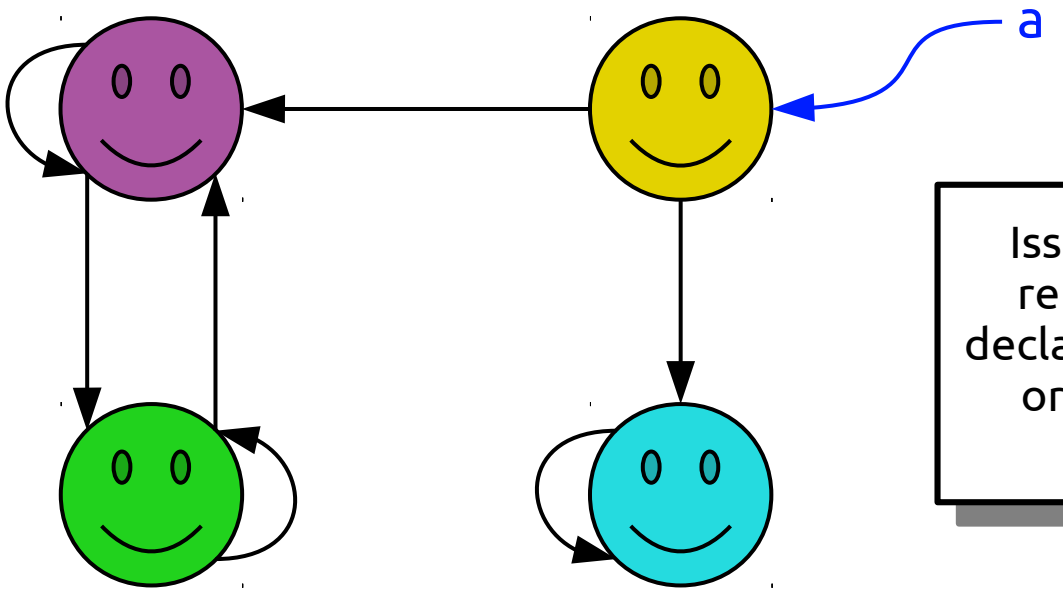
Reflexividade Visualizada



$$\forall a \in A.\ aRa$$

(“Cada elemento está relacionado consigo mesmo.”)

Reflexividade Visualizada



Isso significa que R não é reflexivo, uma vez que a declaração lógica de primeira ordem fornecida abaixo não é verdadeira

~~$\forall a \in A.\ aRa$~~

("Cada elemento está relacionado consigo mesmo.")

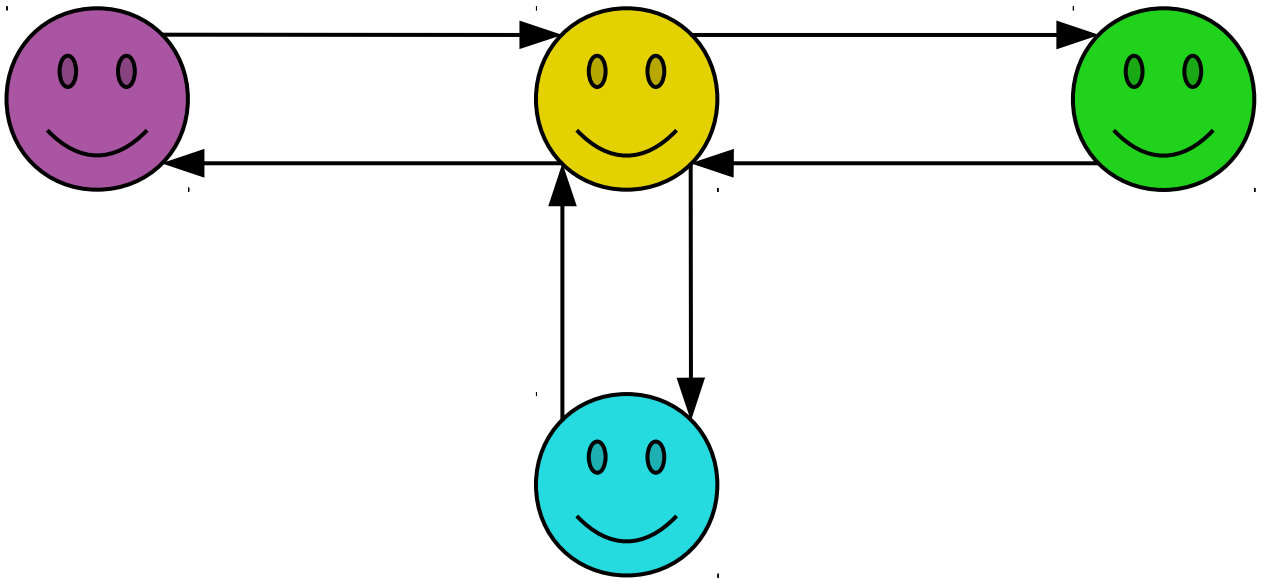
Simetria

- Em algumas relações, a ordem relativa dos objetos não importa.
- Exemplos:
  - Se $x = y$, então $y = x$.
  - Se $x \equiv_k y$, então $y \equiv_k x$.
- Essas relações são chamadas de **simétricas**.
- Formalmente: uma relação binária R sobre um conjunto A é chamada simétrica se a seguinte declaração de primeira ordem for verdadeira sobre R:

  $\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

  ("Se a está relacionado com b, então b está relacionado com a.")

Simetria Visualizada



$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow bRa)$$

(“Se a está relacionado com b, então b está relacionado com a.”)

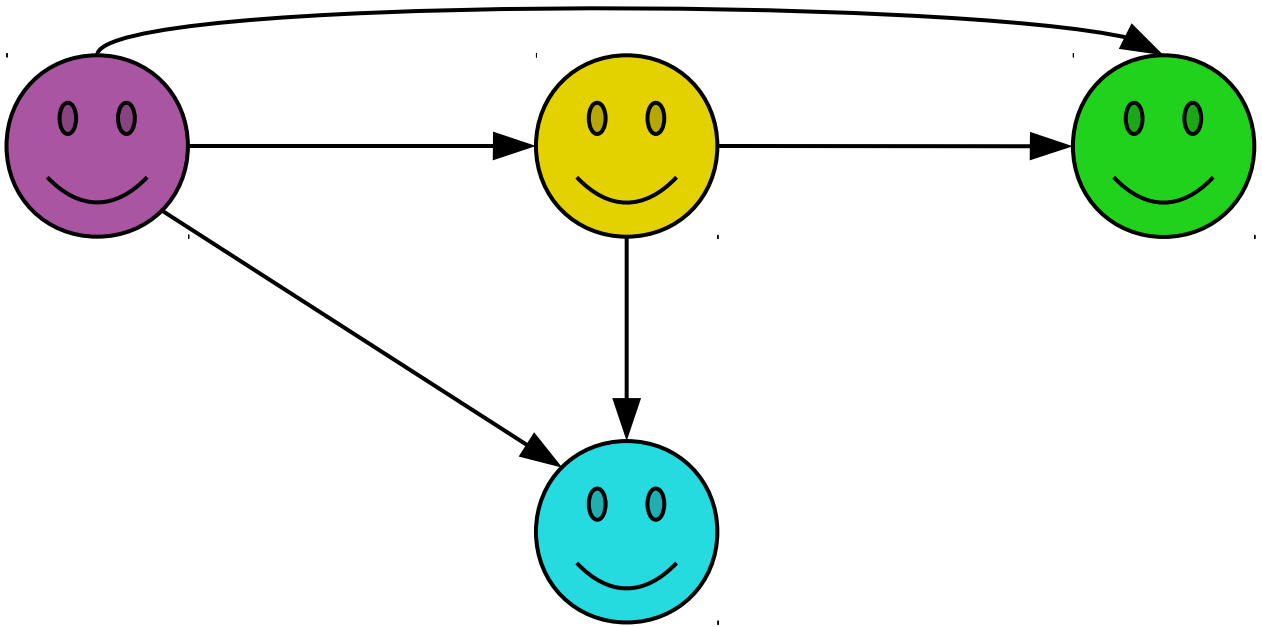
Transitividade

- Muitas relações podem ser encadeadas.
- Exemplos:
  - Se $x = y$ e $y = z$, então $x = z$.
  - Se $R \subseteq S$ e $S \subseteq T$, então $R \subseteq T$.
  - Se $x \equiv_k y$ e $y \equiv_k z$, então $x \equiv_k z$.
- Essas relações são chamadas de transitivas.
- Uma relação binária R sobre um conjunto A é chamada transitiva se a seguinte afirmação de primeira ordem é verdadeira sobre R:

  $\forall a \in A.\ \forall b \in A.\ \forall c \in A.\ (aRb \wedge bRc \rightarrow aRc)$

  ("Sempre que a está relacionado com b e b está relacionado com c, sabemos que a está relacionado com c.")

Transitividade Visualizada



$$\forall a \in A.\ \forall b \in A.\ \forall c \in A.\ (aRb \land bRc \rightarrow aRc)$$

("Sempre que a está relacionado com b e b é relacionado com c, sabemos que a está relacionado com c.")

Relações de Equivalência

- Uma **relação de equivalência** é uma relação reflexiva, simétrica e transitiva.
- Alguns exemplos:
  - $x = y$
  - $x \equiv_k y$
  - x tem a mesma cor que y.
  - x tem a mesma forma de y.

As relações binárias nos fornecem uma linguagem comum para descrever estruturas comuns.

Relações de Equivalência

- A maioria das linguagens de programação modernas inclui algum tipo de estrutura de dados de tabela hash.
  - Java: HashMap
  - C++: std::unordered_map
  - Python: dict
- Se você inserir um par chave/valor e, em seguida, tentar procurar uma chave, a implementação deve ser capaz de dizer se duas chaves são iguais.
- Embora cada linguagem tenha um mecanismo diferente para especificar isso, muitas linguagens os descrevem de maneiras semelhantes...

Relações de Equivalência

- “O método equals implementa uma relação de equivalência em referências de objetos não nulos:
- É reflexivo: para qualquer valor de referência não nulo x, x.equals(x) deve retornar verdadeiro.
- É simétrico: para quaisquer valores de referência não nulos x e y, x.equals(y) deve retornar verdadeiro se e somente se y.equals(x) retornar verdadeiro.
- É transitivo: para quaisquer valores de referência não nulos x, y e z, se x.equals(y) retornar verdadeiro e y.equals(z) retornar verdadeiro, então x.equals(z) deve retornar verdadeiro.”

Documentação Java 8

Relações de Equivalência

“Cada contêiner associativo não ordenado é parametrizado por Key, por um tipo de objeto de função Hash que atende aos requisitos de Hash (17.6.3.4) e atua como uma função hash para valores de argumento do tipo Key e por um predicado binário Pred que induz uma relação de equivalência em valores do tipo Key. Além disso, unordered_map e unordered_multimap associam um tipo T mapeado arbitrariamente com a chave.”

C++14 ISO Spec, §23.2.5/3

Provas de Relação de Equivalência

- Vamos supor que você encontrou uma relação binária R sobre um conjunto A e deseja provar que é uma relação de equivalência.
- Como exatamente você faria isso?

Um Exemplo de Relação

- Considere a relação binária ~ definida sobre o conjunto $\mathbb{Z}$:

  a ~ b se a + b for par

- Alguns exemplos:

  0~4 1~9 2~6 5~5

- Acontece que esta é uma relação de equivalência! Vamos ver como provar isso.

Podemos provar relações binárias, dando uma regra, como esta:

a ~ b se alguma propriedade de a e b for mantida

Este é o modelo geral para definir uma relação. Embora estejamos usando aqui “se” em vez de “se somente”, as duas declarações acima são definicionalmente equivalentes. Por uma variedade de razões, as definições são frequentemente introduzidas com "se" em vez de "se somente".

Quais propriedades devem ter para ser uma relação de equivalência?

Reflexividade

Simetria

Transitividade

Vamos provar cada propriedade de forma independente.

a ~ b se a + b for par

- **Lema 1:** A relação binária ~ é reflexiva.
- **Prova:**

Qual é a definição formal de reflexividade?

$$\forall a \in \mathbb{Z}.\ a \sim a$$

Sendo assim, escolhemos um integer arbitrário a,
então provamos que a ~ a.

a ~ b se a + b for par

- **Lema 1:** A relação binária ~ é reflexiva.
- **Prova:** Considere um $a \in \mathbb{Z}$ arbitrário. Precisamos provar que a ~ a. A partir da definição da relação ~, isso significa que precisamos provar que a + a é par.

  Para ver isso, observe que a + a = 2a, então a soma a + a pode ser escrita como 2k para algum inteiro k (nomeadamente, a), então a + a é par. Portanto, a ~ a é válido, conforme necessário. ■

a ~ b se a + b for par

- **Lema 2:** A relação binária ~ é simétrica.
- **Prova:**

Qual é a definição formal de simetria?

$$\forall a \in \mathbb{Z}.\ \forall b \in \mathbb{Z}.\ (a \sim b \rightarrow b \sim a)$$

Sendo assim, escolhemos integers arbitrários a e b,
onde a ~ b, então provamos que b ~ a.

a ~ b se a + b for par

- **Lema 2:** A relação binária ~ é simétrica.
- **Prova:** Considere quaisquer inteiros a e b onde a ~ b. Precisamos mostrar que b ~ a.

  Como a ~ b, sabemos que a + b é par. Como a + b = b + a, isso significa que b + a é par. Como b + a é par, sabemos que b ~ a, conforme necessário. ■

a ~ b se a + b for par

- **Lema 3:** A relação binária ~ é transitiva.
- **Prova:**

Qual é a definição formal de transitividade?

$$\forall a \in \mathbb{Z}.\ \forall b \in \mathbb{Z}.\ \forall c \in \mathbb{Z}.\ (a \sim b \land b \sim c \rightarrow a \sim c)$$

Sendo assim, escolhemos integers arbitrários a, b e c
onde a ~ b e b ~ c, então provamos que a ~ c.

a ~ b se a + b for par

- **Lema 3:** A relação binária ~ é transitiva.
- Prova: Considere inteiros arbitrários a, b e c, onde a ~ b e b ~ c. Precisamos provar que a ~ c, o que significa que precisamos mostrar que a + c é par.

  Como a ~ b e b ~ c, sabemos que a + b e b + c são pares. Isso significa que existem inteiros k e m onde a + b = 2k e b + c = 2m. Note que

  $$(a+b) + (b+c) = 2k + 2m.$$

  Reorganizando, vemos que

  $$a+c + 2b = 2k + 2m,$$

  Então

  $$a+c = 2k + 2m - 2b = 2(k+m-b).$$

  Portanto, há um inteiro r, nomeadamente, k + m – b, tal que a + c = 2r. Assim, a + c é par, então a ~ c, conforme necessário. ■

Uma Observação

a ~ b se a + b for par

- **Lema 1:** A relação binária ~ é reflexiva.
- **Prova:** Considere um a $\in \mathbb{Z}$ arbitrário. Precisamos provar que a ~ a. A partir da definição da relação ~, isso significa que precisamos provar que a + a é par.

  Para ver isso, observe que a + a = 2a, então a soma a + a pode ser escrita como 2k para algum inteiro k (nomeadamente, a), então a + a é par. Portanto, a ~ a é válido, conforme necessário. ■

A definição formal de reflexividade é dada pela lógica de primeira ordem, mas esta prova não contém nenhum símbolo lógico de primeira ordem!

a ~ b se a + b for par

- **Lema 2:** A relação binária ~ é simétrica.
- **Prova:** Considere quaisquer inteiros a e b onde a ~ b. Precisamos mostrar que b ~ a.

  Como a ~ b, sabemos que a + b é par. Como a + b = b + a, isso significa que b + a é par. Como b + a é par, sabemos que b ~ a, conforme necessário. ■

A definição formal de simetria é dada pela lógica de primeira ordem, mas esta prova não contém nenhum símbolo lógico de primeira ordem!

a ~ b se a + b for par

- **Lema 3:** A relação binária ~ é transitiva.
- Prova: Considere inteiros arbitrários a, b e c, onde a ~ b e b ~ c. Precisamos provar que a ~ c, o que significa que precisamos mostrar que a + c é par.

  Como a ~ b e b ~ c, sabemos que a + b e b + c são pares. Isso significa que existem inteiros k e m onde a + b = 2k e b + c = 2m. Note que

  $(a+b) + (b+c) = 2k + 2m.$

  Reorganizando, vemos que

  $a+c + 2b = 2k + 2m,$

  Então

  $a+c = 2k + 2m - 2b = 2(k+m-b).$

  Portanto, há um inteiro r, nomeadamente, k + m – b, tal que a + c = 2r. Assim, a + c é par, então a ~ c, conforme necessário. ■

A definição formal de transitividade é dada pela lógica de primeira ordem, mas esta prova não contém nenhum símbolo lógico de primeira ordem!

Lógica de Primeira Ordem e Provas

- A lógica de primeira ordem é uma excelente ferramenta para dar definições formais aos termos-chave.
- Embora a lógica de primeira ordem oriente a estrutura das provas, é extremamente raro ver a lógica de primeira ordem em provas escritas.
- Siga o exemplo dessas provas:
  - Use as definições da lógica de primeira ordem para determinar o que assumir e o que provar.
  - Escreva a prova em português simples, usando as convenções que aprendemos em apresentações anteriores.