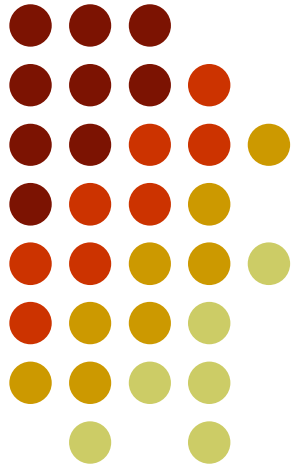
Representação de Conhecimento
Redes Semânticas e Frames

Profa. Josiane M. P. Ferreira e
Prof. Sérgio R. P. da Silva

David Poole, Alan Mackworth e Randy Goebel - "*Computational Intelligence – A logical approach*" - cap. 5
Stuart Russel e Peter Norving - "Inteligência Artificial" - cap. 10
Patrick Henry Winston - "*Artificial Inteligence*" - cap. 2 e cap. 10

agosto/2008

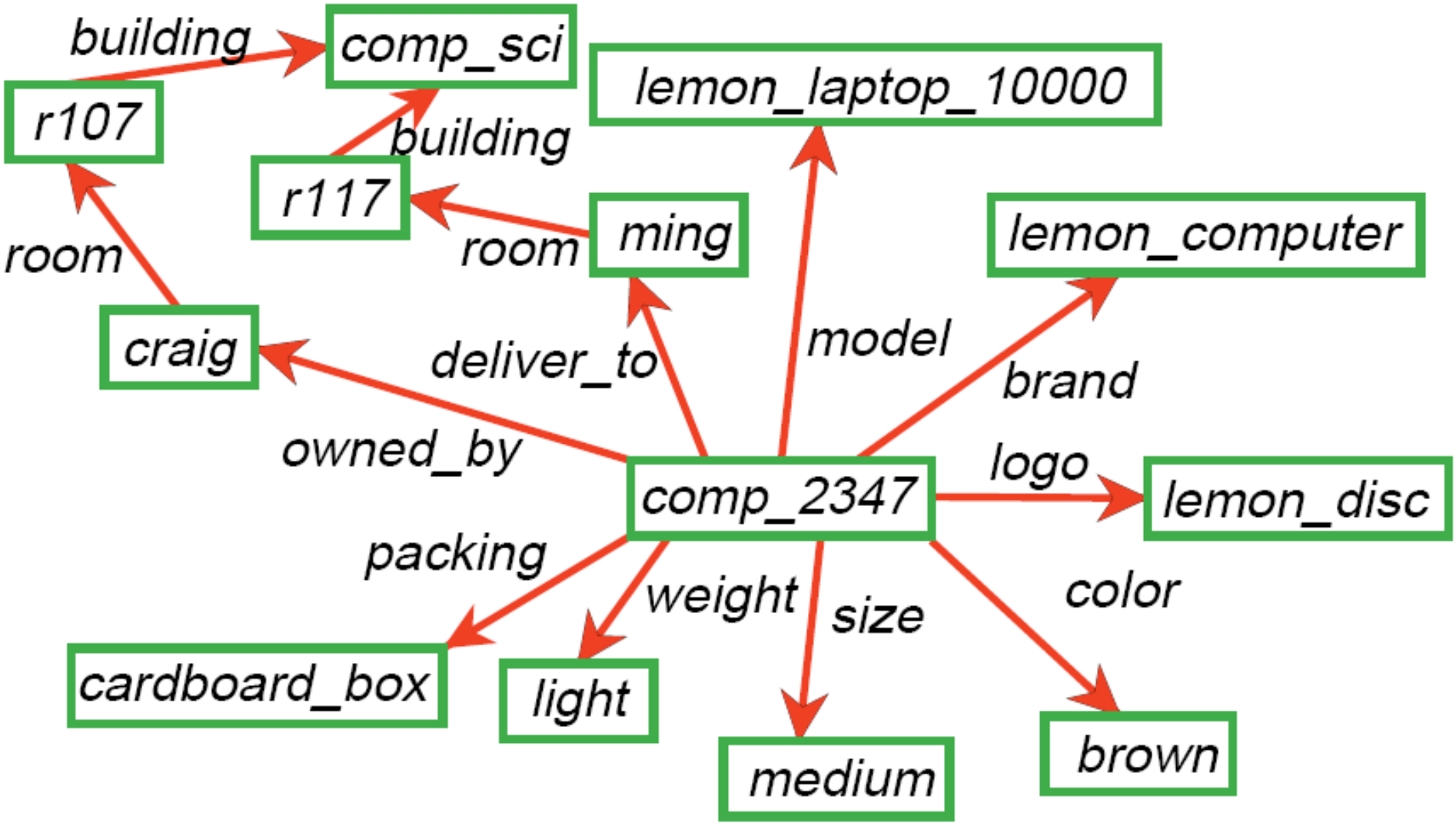


Redes Semânticas

- Quando temos somente uma relação, podemos omiti-la sem qualquer perca de informação
- Assim podemos representar uma relação *prop*(*Obj, Att, Val*) como um grafo onde *Obj* e *Val* são nós e *Att* é o rótulo do arco
- Estes grafos são chamados de **Redes semânticas**

Um exemplo de uma rede semântica

Informações sobre um computador



Programa equivalente em lógica
Informações sobre um computador

$prop(comp\_2347, owned\_by, craig).$

$prop(comp\_2347, deliver\_to, ming).$

$prop(comp\_2347, model, lemon\_laptop\_10000).$

$prop(comp\_2347, brand, lemon\_computer).$

$prop(comp\_2347, logo, lemon\_disc).$

$prop(comp\_2347, color, brown).$

$prop(craig, room, r107).$

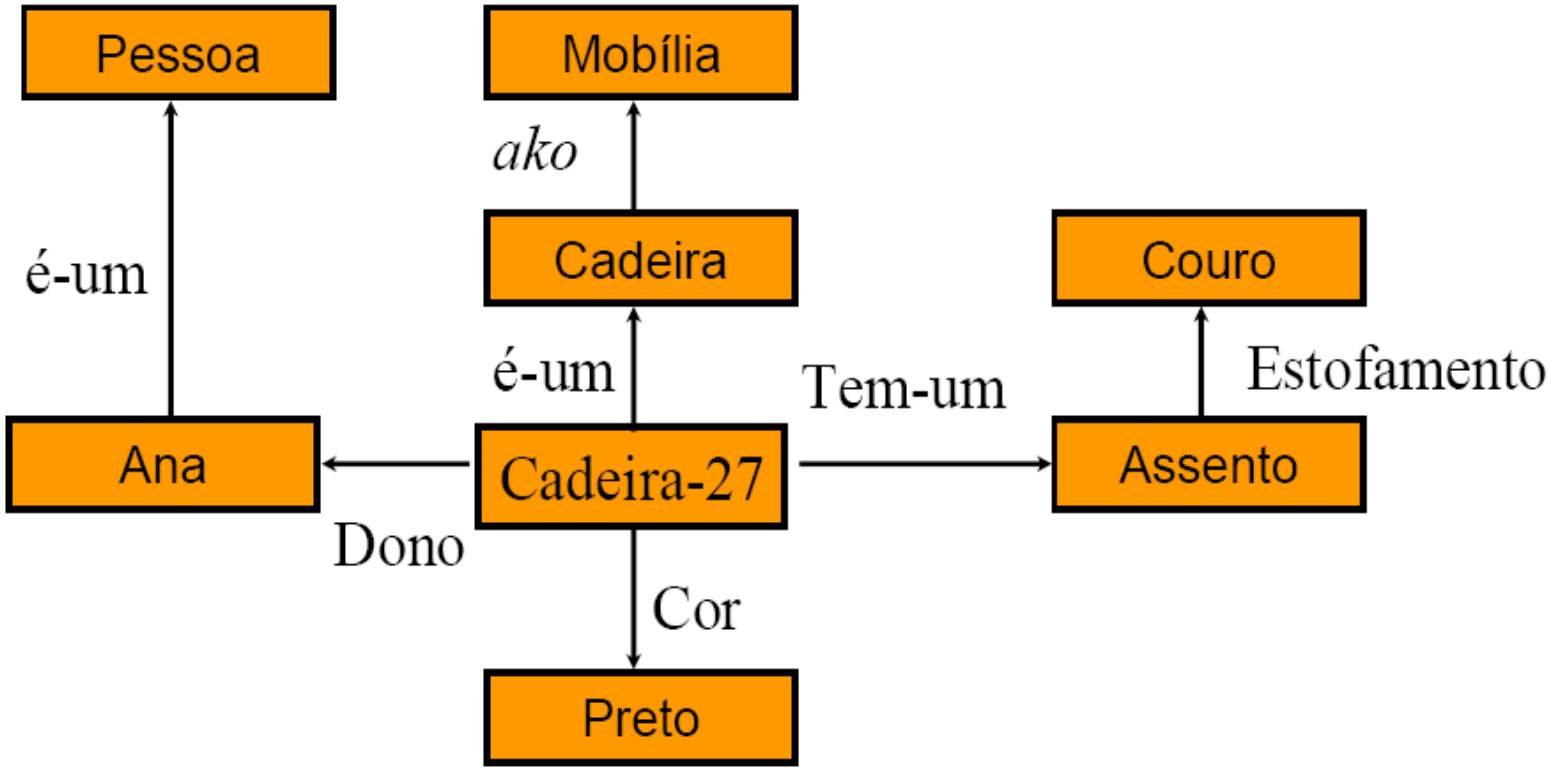
$prop(r107, building, comp\_sci).$

$\vdots$

Redes Semânticas

- Histórico
  - Redes Semânticas foram propostas em 1913 por Selz como uma explicação a fenômenos psicológicos
  - Em 1966, Quillian implementou essas redes e mostrou como o conhecimento semântico poderia ser representado como relacionamento entre dois objetos
- Uma rede semântica é uma representação na qual:
  - Existem nós que representam entidades e *links* (predicados) que representam relacionamentos entre essas entidades;
  - Cada *link* conecta um nó origem até um nó destino;
  - Normalmente, os nós e *links* denotam entidades de um domínio específico.

Redes Semânticas
Mais um exemplo



Redes Semânticas

- Forma mais flexível e intuitiva de representar conhecimento
  - Suportam herança de propriedades
  - Relações entre classes e instâncias
    - **ako (a-kind-of):** relações entre classes
    - **é-um (is-a):** relações entre classes e instâncias
      - Uma classe pertence a uma classe mais alta ou a uma categoria de objetos
  - Relações todo-parte
    - **tem-um (has-a):** identifica características ou atributos das entidades
    - **parte-de (part-of):** identifica características ou atributos das entidades
  - Relações diversas
    - **Variados:** identifica características gerais

Relações em Redes Semânticas

Link Type	Semantics	Example
$A \xrightarrow{Subset} B$	$A \subset B$	$Cats \subset Mammals$
$A \xrightarrow{Member} B$	$A \in B$	$Bill \in Cats$
$A \xrightarrow{R} B$	$R(A, B)$	$Bill \xrightarrow{Age} 12$
$A \xrightarrow{\boxed{R}} B$	$\forall x \ \ x \in A \ \Rightarrow \ R(x, B)$	$Birds \xrightarrow{\boxed{Legs}} 2$
$A \xrightarrow{\boxed{\boxed{R}}} B$	$\forall x \ \exists y \ \ x \in A \ \Rightarrow \ y \in B \wedge R(x, y)$	$Birds \xrightarrow{\boxed{\boxed{Parent}}} Birds$

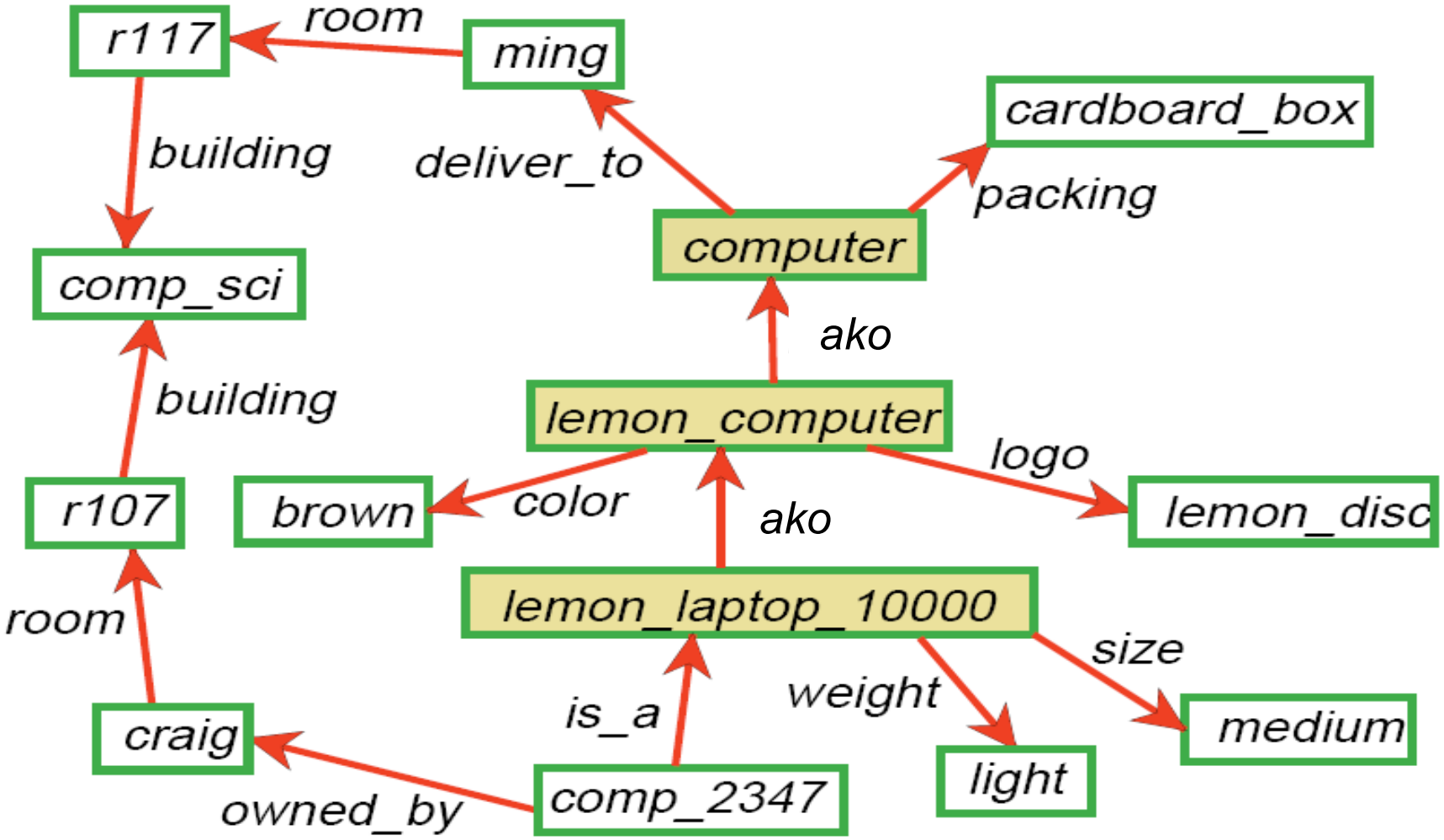
Relações primitivas e derivadas

- **Conhecimento primitivo**: definido explicitamente pelos fatos
- **Conhecimento derivado**: definido pelas regras
  - Permite uma representação mais compacta
  - Permite que conclusões sejam tiradas das observações do domínio
    - Importante porque não observamos tudo diretamente do domínio
    - Muito do que sabemos do domínio é inferido pelas observações e pelo conhecimento mais geral

Relações primitivas e derivadas

- Ex: Ao invés de especificar que o computador comp_2347 tem como logo um lemon disc, podemos saber que todos os computadores da marca Lemon tem este logo
- Associar as propriedades com as **classes** e não com os indivíduos
- Permitir um atributo especial **é_um** entre um indivíduo e uma classe ou **a_kind_of** entre duas classes
  - Permitindo **herança de propriedades**

Uma rede semântica estruturada

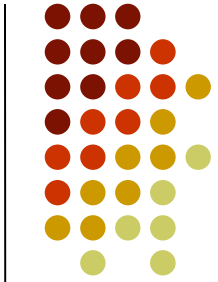


A lógica da herança de propriedade

- Uma arco *p* de uma classe *c* significa que todo indivíduo na classe tem valor *n* para o atributo *p*
  - *prop(Obj, p, n) <-- prop(Obj, é_um, c).*
  - *Exemplo:*

$$prop(X, weight, light) \leftarrow$$
$$prop(X, is\_a, lemon\_laptop\_10000).$$
$$prop(X, is\_a, lemon\_computer) \leftarrow$$
$$prop(X, is\_a, lemon\_laptop\_10000).$$

Herança múltipla



- Um indivíduo normalmente é membro de mais de uma classe
  - Ex: a mesma pessoa pode ser uma mãe, uma professora, uma árbitra de futebol...
- Um indivíduo pode herdar propriedades de todas as classes – **herança múltipla**
- Se existem valores *default* para as propriedades
  - Podemos ter um problema quando um indivíduo herda valores *default* conflitantes de classes diferentes: **problema da herança múltipla**

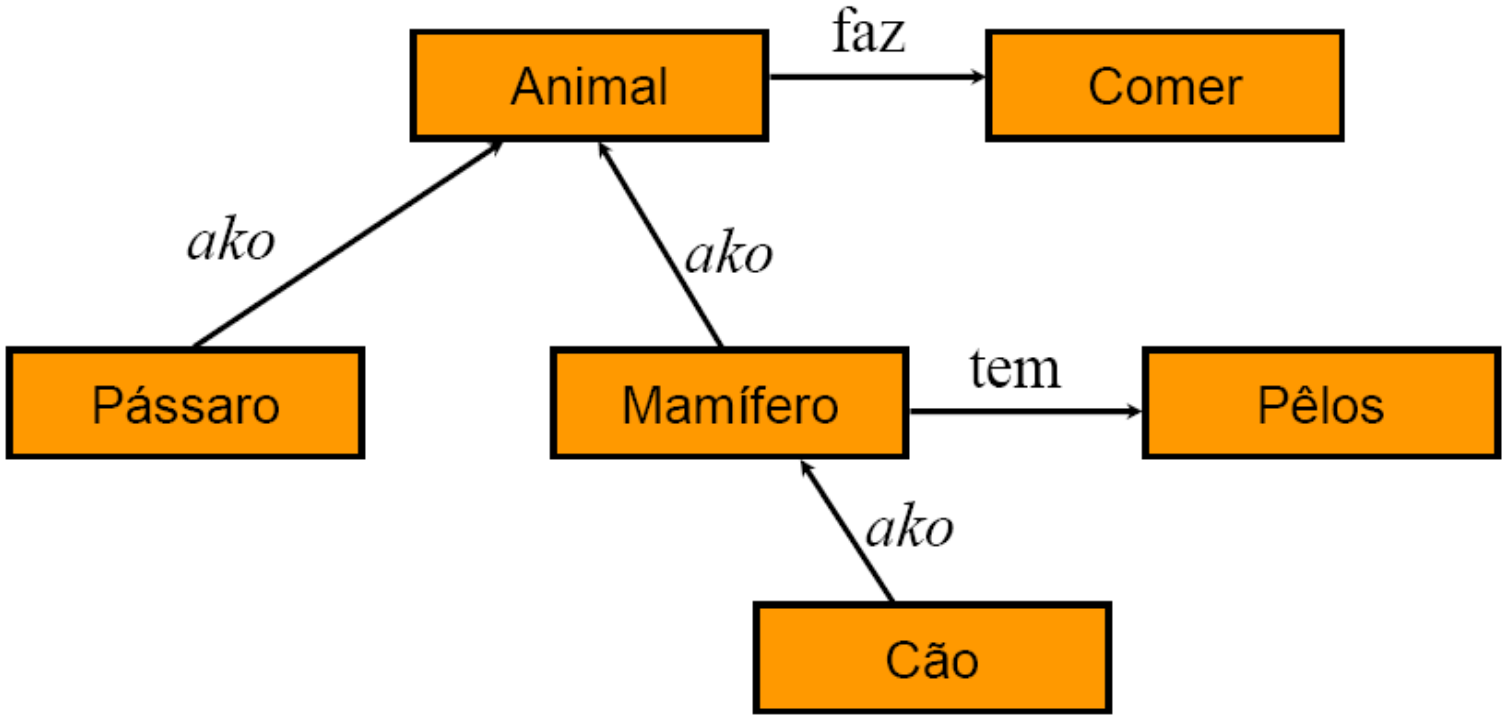
Escolhendo entre relações primitivas e derivadas

- Associar um valor de um atributo com a classe mais geral que contém o valor
- Não associe propriedades contingentes de uma classe com a classe
  - Ex: Pode ser que todos os computadores de uma empresa sejam de uma determinada marca, com gabinete preto.
    - Não é uma boa idéia associar a cor preta a todos os computadores
    - Podem ser comprados computadores de outras marcas com cores diferentes
- Axiomatize na direção **causal**. Queremos conhecimento que seja estável com as mudanças do mundo.
  - a ^ b, a ← b ou b ← a?

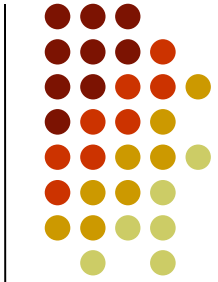
Sistemas de Redes Semânticas

- Base de conhecimento
  - Formada por nós e *links* da rede
- Máquina de inferência
  - Faz busca e casamento de padrões
  - A busca se dá para frente e para trás através dos *links*
- A busca pode ser usada de várias maneiras para se extrair informações
  - Como uma ferramenta explicativa
  - Para explorar exaustivamente um tópico
  - Para encontrar o relacionamento entre dois objetos

Redes Semânticas
Exemplo de busca



Busca como ferramenta explicativa

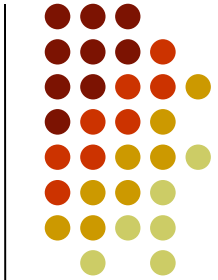


- Para provar a declaração “Cães comem”
  - Pode-se supor que cães comem, e usar busca sobre a rede para provar a hipótese
- Buscando a partir do nó “Cão, temos:
  - “Cão é um mamífero”
  - “Mamífero é um animal”
  - “Animal faz comer”
  - Isto é uma prova para “Cães comem”

Explorar exaustivamente um tópico

- Para derivar todo o conhecimento sobre “cães”, usa-se **busca em largura** a partir do nó “cão”
  - “Cães são mamíferos”
  - “Cães tem pêlos”
  - “Cães são animais”
  - “Cães comem”

Relacionar dois tópicos



- Para verificar se “Cães” e “Pássaros” estão relacionados, pode-se executar, a partir de ambos os nós, uma **busca em largura**
- A interseção entre os nós visitados nos dá o relacionamento entre os nós iniciais
- Isto é chamado **ativação distribuída** ou **interseção de busca**

Redes Semânticas
Vantagens

- Representação visual fácil de entender
  - Os humanos entendem os relacionamentos com facilidade sem necessariamente saber a sintaxe
  - Ajuda os construtores da BC a estruturar seu conhecimento
- Flexibilidade na manipulação de nós e *links*
  - Adição, exclusão e modificação
- Economia escrita e tempo
  - Herança via relações “é-um” e “ako”
- “Capta senso-comum”
  - Semelhante ao armazenamento de informações no cérebro

Redes Semânticas
Limitações

- Busca em redes semânticas grandes pode ser muito ineficiente
- Não há homogeneidade na definição de nós e links
- Hereditariedade pode causar dificuldades no tratamento de exceções
- Pode haver conflito entre características herdadas
- É difícil representar conhecimento procedimental
- Sequenciamento e tempo não são explícitos

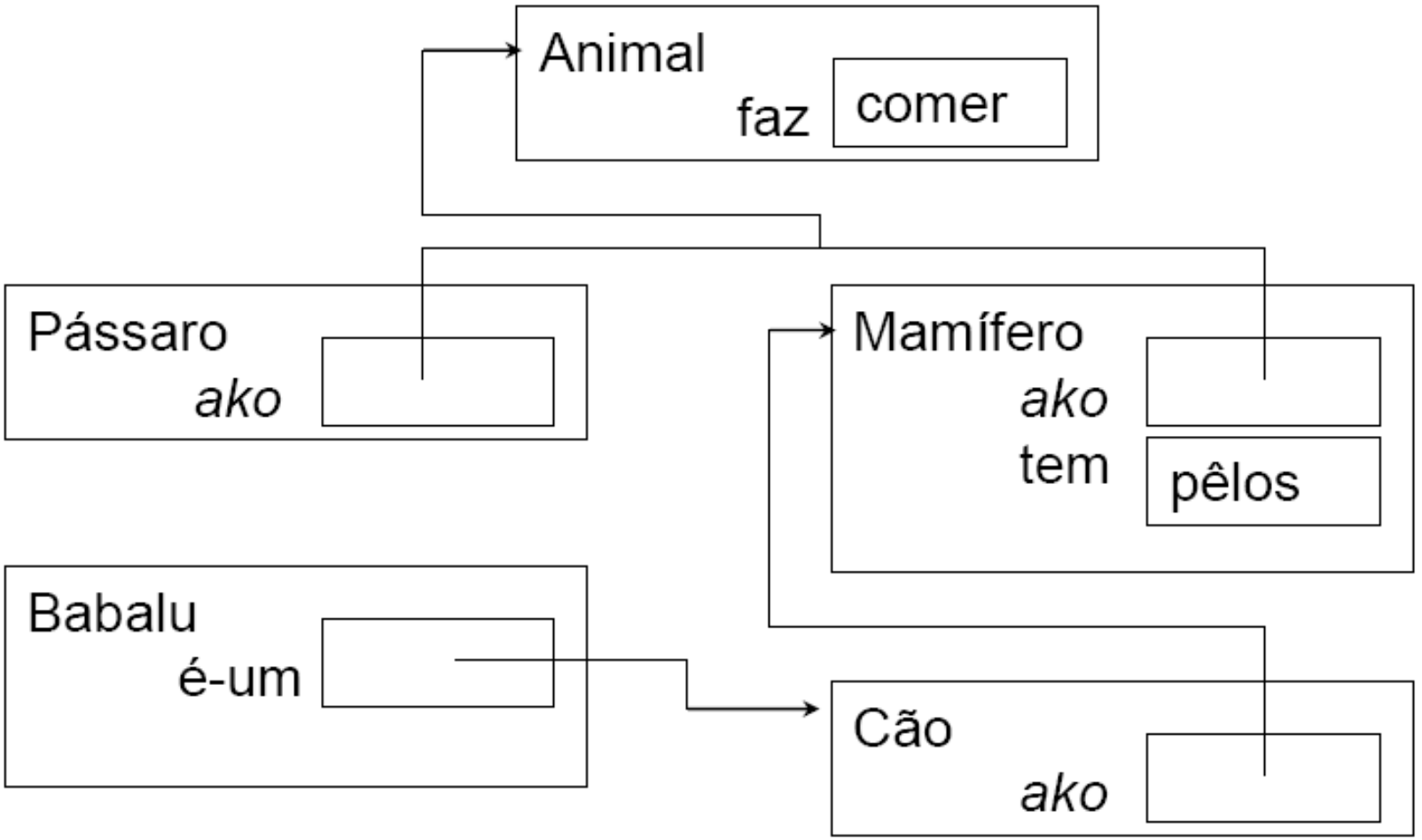
Frames

- Histórico
  - Artigos publicados por Minsky (1974), Winston (1975) Haugeland (1981), Brachman e Levesque (1985)
- Características:
  - Um *frame* é identificado por um nome e descreve um objeto complexo através de um conjunto de atributos.
    - Agrupamos todos os pares atributos-valores das redes semânticas juntos em um simples objeto
  - Um **Sistema de *Frames*** é um conjunto de *frames* organizados hierarquicamente
  - São uma evolução das Redes Semânticas:
    - Nós são substituídos por *frames*.
    - Arcos são substituídos por atributos (*slots*)
    - **Procedimentos podem ser anexados a um *frame*.**

Frames
Slots (atributos)

- *Frames*:
  - Possuem pelo menos dois atributos:
    - *Nome*
    - *ako* ou *is-a*
  - A fim de melhorar a estruturação (hierarquia), privilegiam dois tipos de relações:
    - *ako*: relação entre classe e sub-classe.
    - *is-a*: relação entre classe e instância.
- Cada atributo:
  - Aponta para um outro *frame* ou para um tipo primitivo, ex. *string*;
  - Consiste em um conjunto de **facetas** (atributos de atributos).

Exemplo de um sistema com vários frames



Frames

- Um exemplo de um frame para um computador como uma lista de atributos e valores (***fillers***)

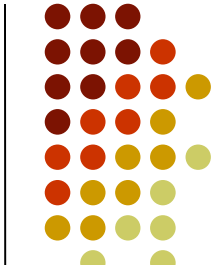
$$
\begin{aligned}
&[owned\_by = craig,\\
&deliver\_to = ming,\\
&model = lemon\_laptop\_10000,\\
&brand = lemon\_computer,\\
&logo = lemon\_disc,\\
&color = brown,\\
&\ldots]
\end{aligned}
$$

- Podemos definir uma *template* de um *frame* para cada tipo de indivíduo
  - Especifica quais slots são necessários e quais são opcionais para um indivíduo daquele tipo
  - Ajuda a construir a BC

Facetas (*facets*)

- Descrevem conhecimento ou algum procedimento relativo ao atributo
- Propriedades:
  - **Valor**: especifica o único valor possível.
  - **Valor default:** especifica o valor assumido pelo atributo caso não haja nenhuma informação a esse respeito.
  - **Tipo**: indica o tipo de dado do valor.
  - **Domínio**: descreve os valores possíveis para o atributo.
  - **Procedimentos - *Demons***
    - Funcionam como os *triggers* nos bancos de dados.

Uma representação abstrata de um frame



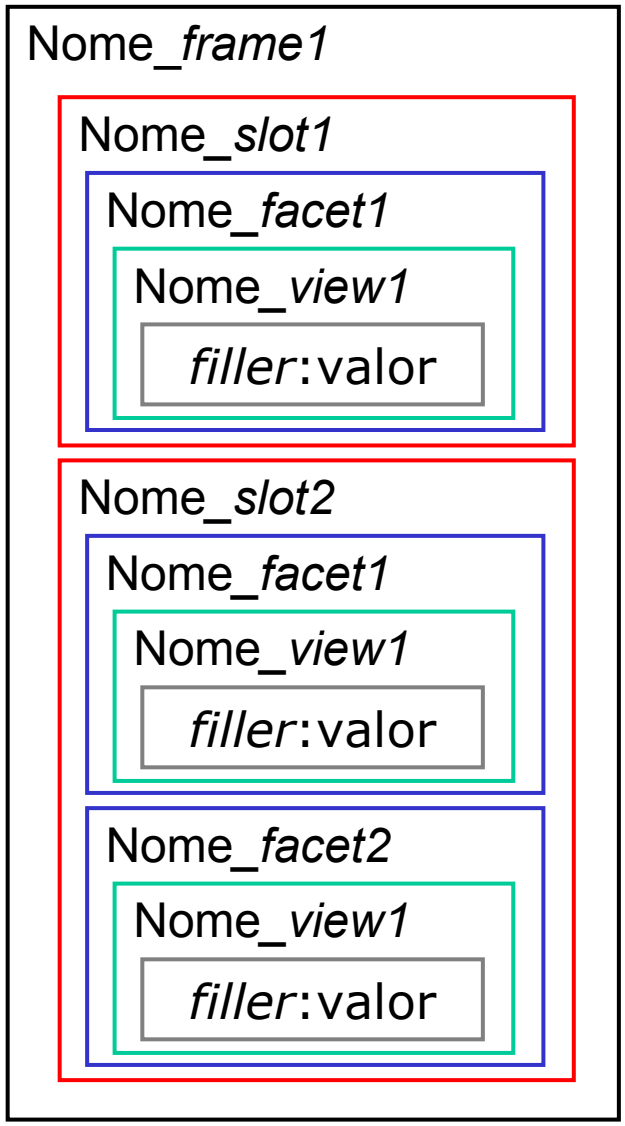
- <Nome do *Frame*>
  - <atributo$_1$>
    - <faceta$_1$>: valor
  - <atributo$_2$>
    - <faceta$_1$>: valor
    - <faceta$_2$>: valor
    - <faceta$_3$>: valor
  - <atributo$_3$>
    - <faceta$_1$>: valor
    - <faceta$_2$>: valor
    - <faceta$_3$>: valor

- Os *frames* integram conhecimento **declarativo** sobre objetos e eventos e conhecimento **procedimental** sobre como recuperar informações ou calcular valores.

Um Frame pode ter vários níveis

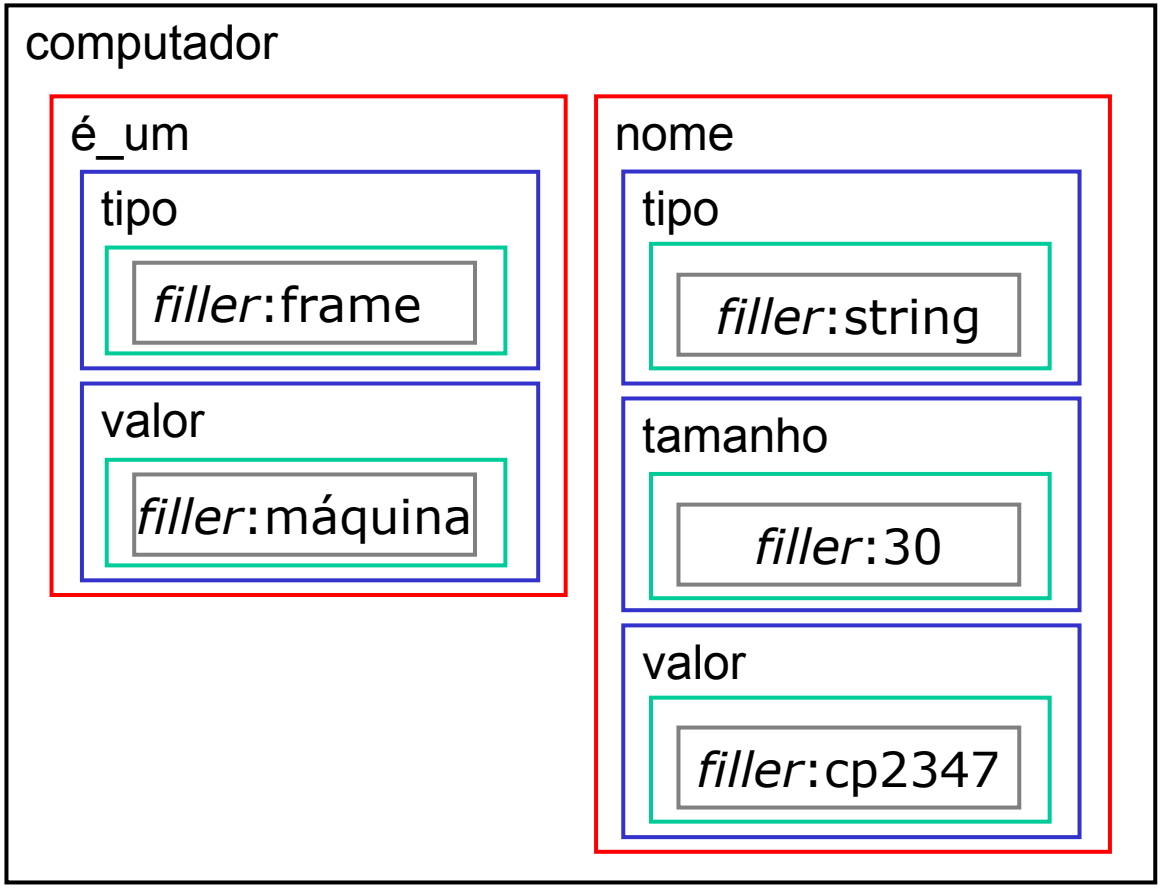
- É formado por um conjunto de ***slots***, que representam atributos dos objetos
- *Slots* podem possuir várias ***facets***, que trazem informações particulares de cada *slot* (atributo)
  - Exemplo: faixa de valores possíveis para o *slot*, ou forma de calculá-los
- Uma *facet* pode possuir várias ***views***, que representam diferentes contextos em que aquela *facet* pode ser avaliada
- Cada *view* possui um ***filler***, ou seja o valor daquela *view*, para aquela *facet*, para aquele *slot*
- Podemos ter *deamons* (procedimentos) associados a *slots*
  - Ex: se o valor de determinada *facet* é x, crie a *facet* y e z.

Exemplo: um Frame de vários níveis




```
Frame1[
   slot1[
      facet1[view1[valor]]
   ],
   slot2[
      facet1[view1[valor]],
      facet2[view1[valor]]
   ]
]
```

Uma visão gráfica de um frame de vários níveis



Um frame de vários níveis representado como listas em Prolog

```
computador[
   é_um[
      tipo[ [ frame]],
      valor[ [ máquina]]
   ]
   nome[
      tipo[ [ string]],
      tamanho[ [ 30]],
      valor[ [ cp2347]]
   ]
]
```

Procedimentos (*Deamons*)

- Definição:
  - São procedimentos **anexados** aos *frames*, disparados por consultas ou atualizações.
  - Podem inferir valores para atributos a partir de valores de outros atributos especificados anteriormente em qualquer *frame* do sistema.
- Procedimentos - *Deamons*:
  - *when-requested*
    - Quando o valor é pedido mas não existe ainda.
  - *when-read*
    - Quando valor é lido.
  - *when-written*
    - Quando valor é modificado.

Exemplo de Procedimentos (Deamons)

Cômodo	*ako*: Lugar-coberto		
Atributo	*Default*	Tipo	*when-needed*
Nº de paredes	4	número	
Formato	retangular	símbolo	
Altura	3	número	
Área		número	
Volume		número	Área * Altura

ako

Sala	*ako*: Cômodo	
Atributo	*Default*	Tipo
Mobiliário	(sofá,mesa,cadeiras)	lista de símbolos
Finalidade	convivência	símbolo
Área	25	número

Herança de propriedades

- Três tipos de informações podem ser de herdadas.
  - valor (= POO)
  - procedimento (= POO)
  - valor *default*
- Idéia: herdar das classes superiores.
  - Em caso de conflito, vale a informação mais específica.
- Existem dois tipos de herança:
  - **Herança simples**
    - Existe uma única super-classe para cada classe
  - **Herança múltipla**
    - Uma classe pode ter mais de uma super-classe, podendo herdar propriedades ao longo de diversos caminhos diferentes

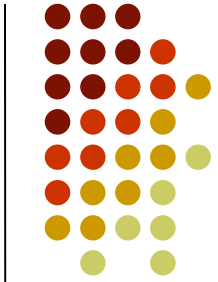
Sistemas de Frames
Funções

- Reconhecer que uma dada situação pertence a uma certa categoria (*matching --> subsunção e classificação*).
  - ex. reconhecimento visual de uma sala de aula
- Interpretar a situação e/ou prever o que surgirá em termos da categoria reconhecida (*matching*).
  - ex. pessoa com revolver (revolver arma → perigo)
- Capturar propriedades de **senso comum** sobre pessoas, eventos e ações.
  - Foi a primeira tentativa de estruturar conhecimento declarativo sem usar regras.
  - Deu origem ao que chamamos hoje de **Ontologias**!

Compartilhamento de conhecimento

- Se mais de uma pessoa está construindo a base de conhecimento, elas devem ser capazes de compartilhar a conceitualização.
- Uma **conceitualização** é um mapa do domínio de problema para uma representação. Uma conceitualização especifica:
  - Que tipos de objetos estão sendo modelados
  - O vocabulário para especificar objetos, relações e atributos
  - O significado ou intenção das relações ou atributos
- Uma **ontologia** é uma especificação formal de uma conceitualização.

Web Semântica



- Ontologias são publicadas na *web* de forma legível para máquinas e são publicamente legíveis.
  - Construtores de bases de conhecimento ou *websites* aderem e se referem a uma ontologia publicada:
    - O mesmo símbolo significa a mesma coisa em vários *websites* que obedecem a ontologia.
    - Se alguém quer se referir a algum outro objeto ou relação, a ontologia é expandida. A comunidade precisa concordar em uma nova terminologia.

Desafios para construir ontologias

- Elas pode ser muito grandes: achar uma terminologia apropriada para um conceito pode ser difícil.
- Como alguém divide o mundo pode depender da aplicação.
- Ontologias diferentes descrevem o mundo de diferentes formas.
- As pessoas podem fundamentalmente discordar sobre a estrutura apropriada.
- Bases de conhecimento diferentes podem usar ontologias diferentes.

Desafios para construir ontologias

- Para possibilitar que BC baseada em ontologias diferentes interoperem, deve haver um mapeamento entre as diferentes ontologias.
- É necessário que seja do interesse do usuário em usar a ontologia.
- O computador não pode entende o significado dos símbolos. O formalismo pode restringir o significado, mas não pode defini-lo.