Chapter 6
Real-World Knowledge Representation and Reasoning

Abstract Apart from its theoretical significance, the AI must represent real-world knowledge, and produce reasoning using that. The real-world things are collections of entities in different classes. This chapter presents the representations structures for such knowledge, e.g., taxonomies and reasoning based on that. Other phenomena in real-world, that are presented are, action and change, commonsense reasoning, ontology structures for different domains, like, language, and world. The Sowa's ontology for objects, and processes, both concrete and abstract, is explained. The situation calculus is presented in its formal details, along with worked exercises. The more prevalent real-world reasoning, like nonmonotonic and default reasoning are also treated in sufficient details, along with supporting worked exercises. This is followed, with summary of the chapter, and exhaustive list of practice exercises.

Keywords Real-world knowledge · Ontologies · Ontological reasoning · Sowa's ontologies · Situation calculus · Nonmonotonics reasoning · Default reasoning

6.1 Introduction

Who is taller, the father or his son held in his hands by the father? Can you make tea out of salt? If you push a needle into a potato, does the needle make a hole in the potato or in the needle? We may find these questions as absurd, but many tasks, such as machine vision, natural language processing, planning, and reasoning in general, requires the same kind of real-world knowledge and capabilities of reasoning. As another example, if a six feet tall person is holding a two feet baby in his arms, and it is also given that they are father and son, we take no time to conclude which one is father and which one is son.

The use of real-world knowledge for natural language processing (NLP), particularly for all kinds of sense disambiguation, and for machine translation is a challenging task. Some of the ambiguities are resolved using the simple rules, while the others can be resolved using rich understanding of the world. Consider the following two examples:

K. R. Chowdhary, *Fundamentals of Artificial Intelligence*,
https://doi.org/10.1007/978-81-322-3972-7_6

"The Municipal council of the city refused permission to demonstrators because they feared violence," versus,

"The Municipal council of the city refused permission to demonstrators because they advocated violence."

In the first sentence it is required to determine that "they" refers to "Municipal council of the city" and not to "demonstrators." But, in the second sentence it is required to find out that, "they" refers to "demonstrators." The above determination requires the knowledge about the characteristic relations of "Municipal council of city" and "demonstrators" with the verbs "fear" and "advocated." In NLP, such ambiguities in the language can be resolved by true understanding of the text, which requires bringing of real-world knowledge in the system.

Considering computer vision, we are able to recognize the objects because of context. For example, the *bowl*, *banana*, *knife* in kitchen are for container, for eating, and to cut vegetables, respectively. But, the same items in a departmental store are for sale. So, it is relation with other objects/environment, like kitchen and shop, explain the meaning of objects [1].

Learning Outcomes of this Chapter:

1. Compare and contrast the most common models used for structured knowledge representation, highlighting their strengths and weaknesses. [Assessment]
2. Identify the components of non-monotonic reasoning and its usefulness as a representational mechanism for belief systems. [Familiarity]
3. Compare and contrast the basic techniques for representing uncertainty. [Assessment]
4. Compare and contrast the basic techniques for qualitative representation. [Assessment]
5. Apply situation and event calculus to problems of action and change. [Usage]
6. Explain the distinction between temporal and spatial reasoning, and how they interrelate. [Familiarity]

6.2 Taxonomic Reasoning

Taxonomy is a name given to collection of individuals, categories, and the relations between their pairs. The taxonomies are also referred as *semantic networks*. As an example, Fig. 6.1 shows a taxonomy of some categories of animals, individuals, and the relations between categories and individuals, as well between category and its subcategories.

There are three basic relations types [2]:

- A category is a subset of other category. For example, the categories *dog* and *cat* are subsets of another category *mammal*;

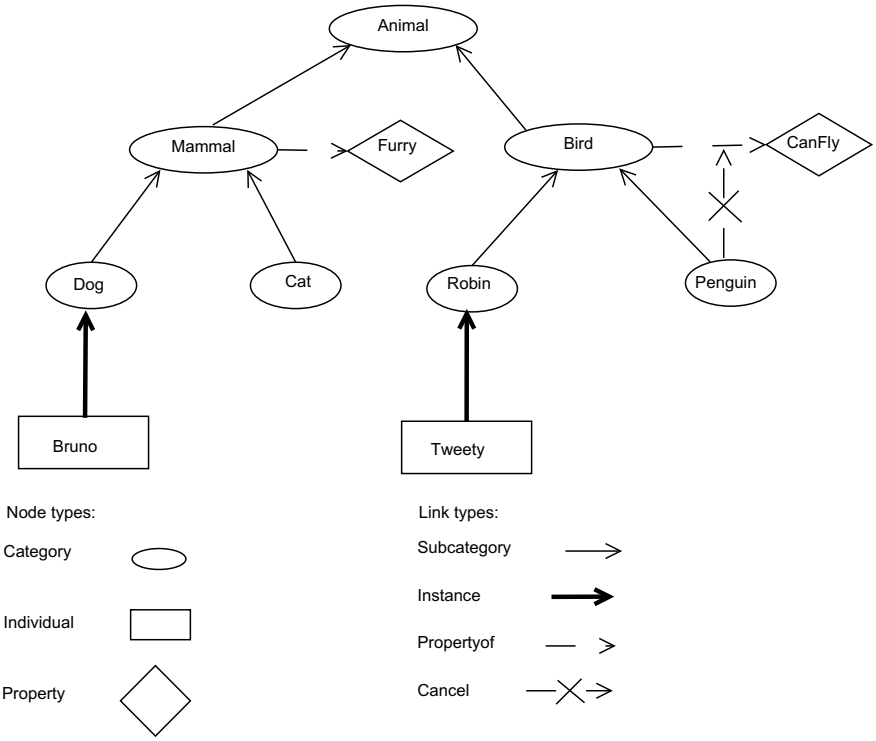

Fig. 6.1 Taxonomy of animal world

- An individual can be an instance of a category. For example, an individual called *Bruno* is an instance of a category *dog*; *robin* and *penguin* are subsets (subcategories) of *bird*;
- Two categories are always disjoint. For example, *dog* and *cat* are disjoint sets.

A property can be used a tag for a category. For example a category *Mammal* can be tagged with property *furry*, and another category *bird* can be tagged with property *Canfly*.

One form of inference from taxonomical structure is through the relation of *transitivity*. In the taxonomy shown in Fig. 6.1, Bruno is shown as an instance of dog, and dog is a subset of mammals, then it can be reasoned that Bruno is an instance of mammal.

The other form of inference—the *inheritance*, works as follows:

Bruno is an instance of dog,
Dog is subset of mammal,
Mammal has property *furry*,
Therefore, it can be inferred through inheritance that dog and Bruno have property *furry*.

A variant of this inheritance is *default inheritance*, which have following characteristics:

1. A category can be marked with a characteristic but cannot be marked as a universal property, and
2. A subcategory or instance will always inherit property from higher order category unless otherwise it is specifically negated.

For instance, there is a category *bird*, with property *Canfly*. And, there are subcategories Robin and Penguin. The property *Canfly* should be inherited by sub-category Robin, but not by Penguin. This is implemented by negating the Canfly property for Penguin, as shown in the simple and standard taxonomy of the animal world in Fig. 6.1, where every two categories are related in such a way that, they are either disjoint or one is subcategory of other.

Other taxonomies structures are less straight forward. Consider a semantic network for categories of people, the individual say *Dr. C. V. Raman*, is having following relations with other categories, as an instance of those categories. He is simultaneously a *Physicist*, a *Nobel-laureate*, a *ScientistIndian*, and *NativeofMadras*. Also, there is an overlap, and it is some times not clear as which are the properties, and which are taxonomic categories. Hence, in taxonomizing more abstract categories, choosing and delimiting the categories become a challenging task.

Number of specialized taxonomies have been developed in the domains such as medicine, geonomics, languages, and other fields. The more sophisticated taxonomies are based on *description logic*, which provides the tractable constructs for describing concepts and the relation between them. These have been applied in semantic OWL (*Web Ontology Language*).

Temporal Reasoning

It provides the representation of knowledge and automating reasoning about *time*, *duration*, and *time intervals*. For example, in the sentence, "After landing at Frankfurt, it took a long time, before I could board the connecting flight to New Delhi" versus "It for the two countries a long time before they could arrive to trade treaty." In the first sentence, "long time" may mean few hours, while in the second same phrase may mean many years, or even decades. Integrating such reasoning, with natural language (NL) interpretation, has been problematic. Many temporal relations are not explicitly stated in text, and they are complex and text dependent.

Action and Change

Theory of *action and change* is other area of commonsense reasoning, and has been well understood. Following are the constraints of representation and reasoning for the domains of *a*ction and *change*:

- Events are *atomic*, that is, at a time only one event occurs. The reasoning will take place based on the state of the world at the beginning, and at the end of the event. The intermediate states while the event is in execution are not accounted for.
- Each and every change of the world is due to occurrence of some event.

- All the events are deterministic in nature, i.e., state of the world at the end of every event can be fully determined using the state before that event, plus this event itself.

The problem of representation, and the form of reasoning such as prediction and planning, have been largely understood for the domain that satisfy the above mentioned three constraints. However, there are other extreme domains: *continuous domains, simultaneous events, probabilistic events*, and *imperfect knowledge domains*, where, the representation and reasoning are not so well understood yet. Some of these domains will be discussed in the following text.

6.3 Techniques for Commonsense Reasoning

The field of commonsense reasoning can be largely divided into two areas:

- *knowledge based* approaches, and
- *machine learning* approaches that cover large data corpora, which are almost always of the text corpora types.

However, there is a very limited interaction between these approaches. There are other approaches, e.g., *crowd-sourcing* based approaches, which try to construct knowledge base as combination of other knowledge bases and participation of many non-experts [2].

The knowledge based approaches are further divided into categories that are based on following domains (see Fig. 6.2).

1. Mathematical logic or some other mathematical formalism,
2. Informal approaches, which are in contrary to mathematical formalism,
3. Based on theories from cognition, and
4. There are large-scale other approaches, which are usually mathematical or informal, but mainly targeted at collecting a large amount of knowledge.

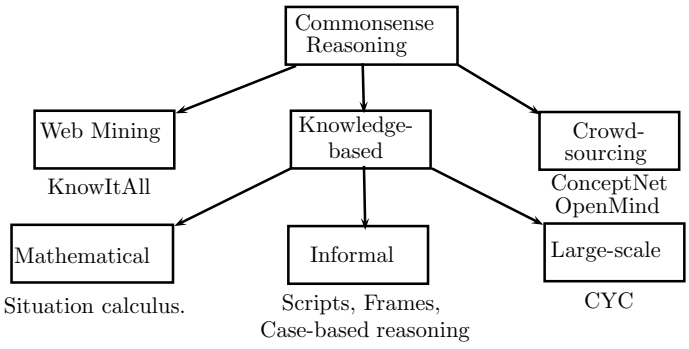

Fig. 6.2 Taxonomy of approaches to commonsense reasoning

One of the successful form of a commonsense reasoning, which has mathematical base is *qualitative reasoning* (QR). The QR helps in automating the reasoning and problem solving about the physical world around us. It has also generated techniques and systems that are being commonly used in application domains, like autonomous support for spacecraft, on-board diagnosis of vehicular systems and their failure analysis, automated generation of control software for photocopiers, and intelligent aids for learning about thermodynamic cycles. There are number of prominent features that are typical for QR systems: ontologies, causality, and inference of behavior from structure.

The other type of reasoning is *quantitative reasoning*, which deals with quantities, and reasons on the criteria of high/low, big/small, long/short etc.

The commonsense reasoning targets a number of different objectives, some of these are as follows:

- *Plausible inference*. It is used for drawing provisional/uncertain conclusions.
- *Reasoning architecture*. The reasoning architecture is a general-purpose data structures to encode knowledge and algorithms, which are helpful for carrying out the reasoning. For example, to represent the meaning/senses of natural language sentences.
- *Basic domains*. While performing the commonsense reasoning, human can do complex reasoning about basic domains like space, time, basic physics, and basic psychology. The knowledge we as human draw from these domains is largely not fully documented and the reasoning processes are largely unavailable for introspection. An automated reasoning should comprise all these reasoning capabilities.
- *Reasoning modes*. The commonsense reasoning system should incorporate a variety of modes of inference, like, explanation-based reasoning, generalization, abstraction, analogical-based reasoning, as well be able to perform the simulation.
- *Independence of experts*. Hand-coding of a large knowledge base for commonsense reasoning is expensive and slow process. Hence, assembling it either automatically or building it by drawing the knowledge of non-experts, e.g., through crowdsourcing, is considered as better choice.
- *Breadth*. To perform a powerful commonsense reasoning through machines, will require a large body of knowledge.
- *Cognitive modeling*. These are the theories of automated commonsense reasoning that describe the commonsense reasoning in people.
- *Applications*. To be useful, a commonsense reasoner must interface with applications smoothly, and must serve the needs of applications.

6.4 Ontologies

Ontology is a particular theory about the nature of being or the kinds of existent. The task of intelligent systems in computer science is to formally represent these existents. A body of formally represented knowledge is based on conceptualization.

A concept is some thing we can form a mental image of. Conceptualization consists of a set of objects, concepts, and other entities about which knowledge is being expressed and of relationships that hold among them. Every knowledge model is committed to some conceptualization, implicitly or explicitly [3].

The major applications of ontologies are in natural language processing, information retrieval, modeling and simulation. The CYC (enCYClopedia) ontology, for example, is used for a CYC natural language system, built for the purpose of translating natural language text into cycL [4].

An ontology is a systematic arrangement of all of the important categories of objects or concepts that exist in some field of discourse, that shows the relations between these concepts/objects. In a completed form, an ontology is a categorization of all of the concepts in some field of knowledge, that includes all the objects, their properties, relations, and functions needed to define the objects, and specify their actions. A simplified ontology may contain only a hierarchical classification of objects and processes (called taxonomy), which shows the *subsumption* relations between concepts in the specific domain of knowledge [5].

An ontology may be visualized as an abstract graph with nodes representing the objects and labeled arcs representing the relations between them, as shown in Fig. 6.3, which is the upper level of the CYC (from en*CYC*lopedia) project hierarchy.

The CYC Project is an example of ontology. CYC contains more than 10,000 concept types used in the rules and facts encoded in the knowledge base. Its goal was to build a commonsense knowledge base containing millions of facts and their interrelationships. It was the efforts to build the knowledge in it that is seldom written down—the knowledge for example, the reader of an encyclopedia is assumed to possess to understand the contents of encyclopedia. This kind of knowledge is required as built-in in a system, which are required to read the restricted natural language.

A concept is some thing we can form a mental image of. At the top of the hierarchy is the *Thing* "concept", which does not have any properties of its own. The hierarchy under *Thing* is quite tangled. Not all the subcategories are exclusive. In general, Thing is partitioned in three ways: First is *Represented Thing* versus *Internal Machine Thing*. Every CYC category must be an instance of one and only one of these sets. Internal-Machine-Thing is anything that is local to the platform CYC is running on (strings, numbers, and so on). *Represented-Thing* is everything else.

The concepts used in an ontology and hierarchical ordering are to some extent arbitrary, and largely depends on the purpose the ontology is created. This is because, same objects are of varying importance when used for different purposes, and one or other set of properties of objects are chosen as the criteria for classification of these objects. Apart from this, different degrees of aggregation of concepts may be used, and the importance for one purpose may be of no concern for a other purpose. For example, *class* is collection of *students*, and when all of them are playing in ground we call it a *team*. As other example, when both parents are teaching they are called teachers, when traveling together they are passengers, and at home they are spouse.

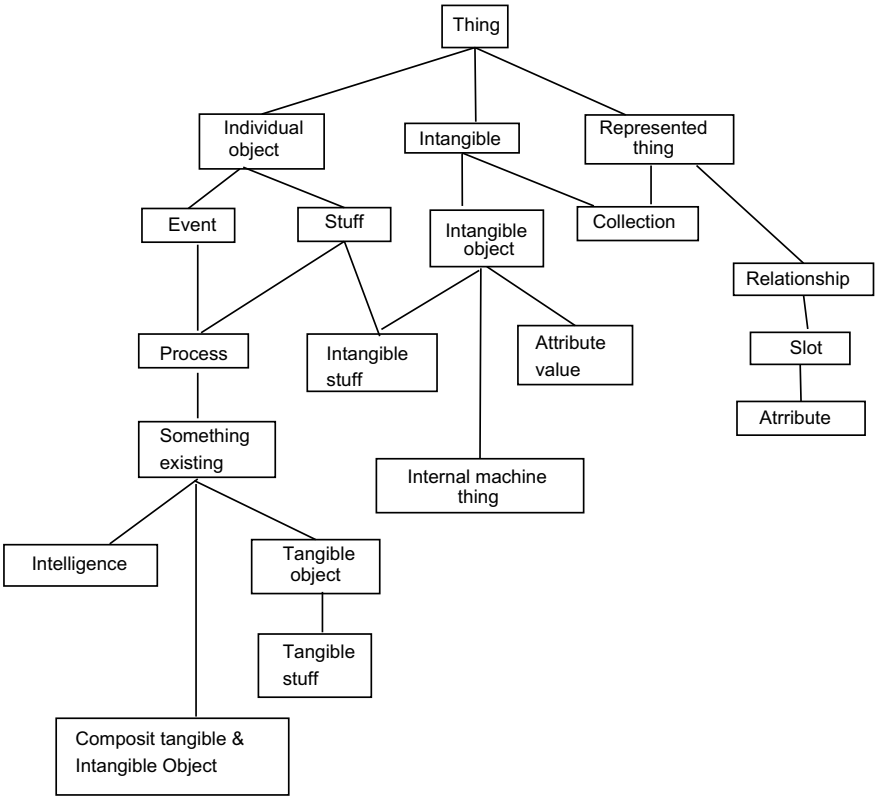

Fig. 6.3 World ontology

6.5 Ontology Structures

When compared with First-Order Predicate Logic (FOPL) for knowledge representation and reasoning, in the ontologies we are particular for knowledge *organization* as well as *contents*. This approach has generalization, combined with exceptions. For example, "all birds fly" can be a rule. But, there are some birds which do not fly, for example, *penguin*. This is an exception. We should be able to add this concepts of knowledge, not as exception, but as extension. This requires categorization of objects.

Ontologies provide explicit representations of domain concepts busing a structure around which knowledge base is built. Each ontology is a structure of concepts and relations between them, such that all concepts are defined as well interpreted using a declarative approach. The ontology system also defines the vocabulary of the problem domain, and a set of constructs as how the terms can be combined to model the domain. The Fig. 6.3 shows a general structure of ontology.

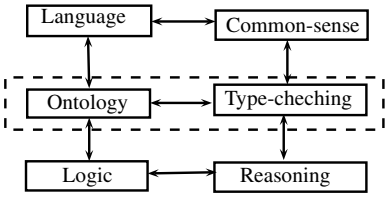

Fig. 6.4 Language, logic, and ontology

6.5.1 Language and Reasoning

The Fig. 6.4 shows the interrelationship of language and logic with ontology, and how each of these are related to commonsense, reasoning, and type checking. The language, ontology, and logic are representations, where as the corresponding blocks, i.e., common-sense, type-checking, reasoning, are respectively, the applications. Consider the object "ball", we identify it as a member in the collection of balls by its type checking; we say "plants have life", they belong to the category of all the living beings (type-check). At lowest level in Fig. 6.4 is *logic*, making use of binary operators. This logic helps in reasoning process. The common-sense is context level knowledge, which is essential for language understanding. For example, the phrase—"a good play", while in a cricket ground is understood as good shot, while in a theater, it is taken as performance by an artist. We resolve the phrase "a good play" by context.

There is a close relationship between natural language understanding (NLU) and knowledge representation and reasoning. In other words, the relationship between *language* and *knowledge*. In fact, understanding is reasoning paradigm, as it has become quite clear that understanding natural language is, for the most part, a commonsense reasoning process at the pragmatic level. As an example, to illustrate this strong interplay between language understanding and commonsense reasoning, consider the following:

(a) "Jack defended a jailed activist in every state."
(b) "Jack knows a jailed activist in every state."

Through commonsense, we have no difficulty in inferring for (a) that Jack supports for the same activist in every state. Perhaps jack might have traveled to different states for campaign. Where as in (b), we hardly conceive of the same. It may mean, "there is a jailed activist in every state", or an activist being jailed in every state. Such inferences lie beyond syntactic and semantic explanations, and are in fact depend on our commonsense knowledge of the world (where we actually live).

As another example, consider resolving of the noun He in the following:

(c) Rajan shot a policeman. He instantly

(i) ran away.
(ii) collapsed.

It is obvious that the above references can only be be resolved through commonsense knowledge. For example, typically, when $shot(x, y)$ holds between some x and some y, the x is the more likely a "subject" which would run away. So, in (c(i)), most likely its is Rajan who ran away. In c(ii), y is the more likely to collapse, that is the policeman. Note, however, that such inferences must always be considered defeasible, since quite often additional information might result in the retraction of previously made inferences. For example, (c(ii)) might be describing a situation in which Rajan, a ten-year old who was shooting for practice, fell down. Similarly, and (c(i)) might actually be describing a situation in which the policeman, upon being slightly injured, tried to run away, may be to escape further injuries!

There are number of challenges, in term of computational complexities, in reasoning with uncommitted (or underspecified) logical forms. However, even bigger challenge is to make available of large body of commonsense knowledge, with computationally effective reasoning engines.

6.5.2 Levels of Ontologies

In comparing various ontologies, they can be viewed at three different levels:

1. *is-a* taxonomy of concepts,
2. the *internal concept* structure and relation between concepts, and
3. the presence or absence of *explicit axioms*.

The Fig. 6.5 shows various concepts related by *is-a* hierarchy relation (a member of relation), also subset relation. A member can be an instance of a category. Taxonomy is central part of most ontologies, and its organization can vary greatly. For example, all concepts can be in one large taxonomy, or there can be number of smaller hierarchies, or there can be no explicit taxonomy at all.

Although all general-purpose ontologies try to categorize the same world, they are very different at the top level. They also differ in their treatment of basic parts: *things*, *processes*, and *relations* (see Fig. 6.3).

The next level of comparison is internal concept structure, which can be realized by *properties* and *roles*. The internal concept relations are the relations, for example, between *bird* versus *canfly, wings, and feathers* (see also Fig.6.5). Concepts in some ontologies are atomic (axioms) and might not have any properties or roles or any other internal structure associated with them.

An important test for any ontology is the practical applications it is used for. These can be the applications in natural language processing, information retrieval, simulation and modeling, and so on, that use knowledge represented in the ontology.

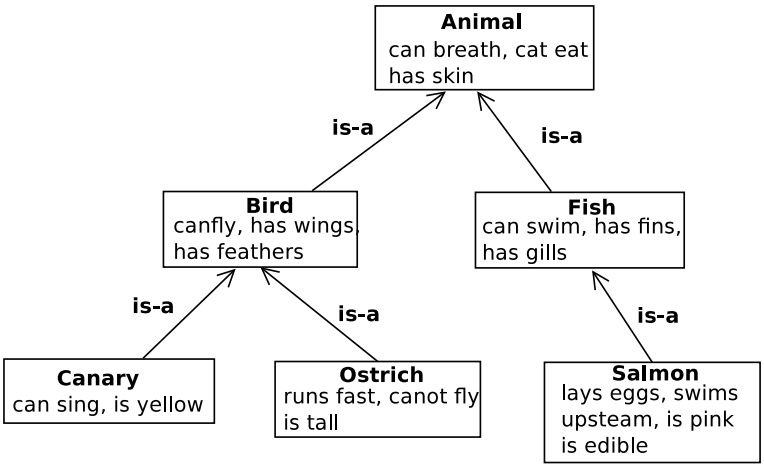

Fig. 6.5 Top level ontology for animal

6.5.3 WordNet

The WordNet is a well developed lexical ontology—a manually constructed online lexical database system, provided with open access and semantically organized lexical objects. Its structure is provided with capability to distinguish between various parts-of-speech: like nouns, adjectives, verbs and adverbs. The basic object of WordNet is *synset*, which is a set of synonyms. Hence, if a word has more than one *senses* (e.g., the word "bank" for "money", and "bank" for "river bank"), it will appear in more than one synset. The synsets are organized in a hierarchy as super-class (hypernyms) and subclass (hyponyms). The Fig. 6.6 shows part of wordnet hierarchy of tangible things, with braces enclosing concepts in same synset. Note that in the synset {living thing, organization}, each of its subordinate item has two parts, e.g., {plant, flora}, but in case of {nonliving thing, object} there is only one item (the object name) in each of the subordinate, because the non-living thing is not classified thing [5, 6].

The Wordnet is a taxonomical architecture, which does not have more elementary concepts or axioms. For each concept (i.e., synset) provided in the Wordnet, there is a pointer to nouns representing its parts. For example, the parts of concept "bird" may be feathers, beak, and tail. There are provisions in the Wordnet for other types of pointers, for example, from noun to verb, to represent functions (actions), or from noun to adjective to represent properties of nouns.

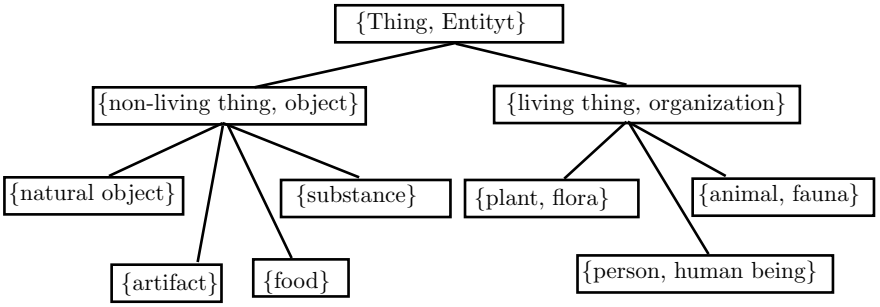

Fig. 6.6 WordNet ontology representing subclass relation among synsets

6.5.4 Axioms and First-Order Logic

Structurally, an ontology is collection of terms, their definitions, and axiom relating them. The terms are typically organized as taxonomy. In some cases axioms are central to ontology design, while in other cases the axiomatic language is associated with central concepts.

Apart from the taxonomy and structure of concepts, axioms are the base for representing more information about categories and their relations. In addition, the axioms specify constraints on properties and role values for each category. Some times, axioms are specifically specified, while other times ontology consists of categories and corresponding frames, and every thing else in hidden in the application code. There is a fine-line difference between internal concept structure and axioms.

The category is represented using a frame formalism, and roles and properties are slot of the frame. The same facts can also be represented using axioms, i.e., a taxonomy can also be represented using axiomatic notations. In this notation, the axioms are represented using first-order predicate logic. For example, "all persons are living being and all living being are things", is represented Fig. 6.6, as well by the following Eq. 6.1.

$$\forall x \forall y[(person(x) \rightarrow livingbeing(x)) \wedge (livingbeing(y) \rightarrow thing(y))] \quad (6.1)$$

The CYC project, for example, comprised among the largest number of axioms, i.e., in the order of 10^6.

6.5.5 Sowa's Ontology

John Sowa (1997, 1995) stated his fundamental principles for ontology design as "distinctions, combinations, and constraints" (see Fig. 6.7). He uses philosophical motivation as the basis for his categorization. There are three top-level distinctions:

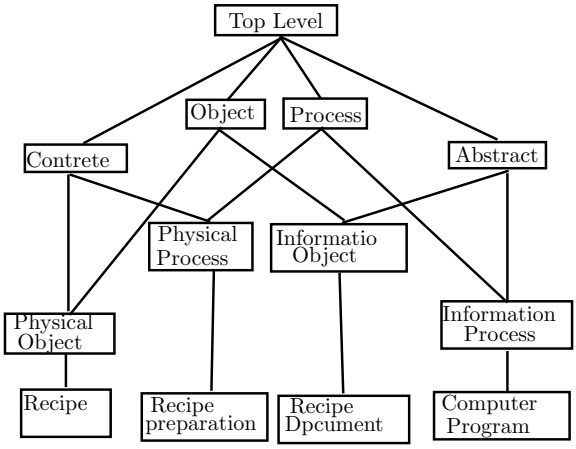

Fig. 6.7 Sowa's ontology

First is *Physical* versus *Information*, or *Concrete* versus *Abstract*. This is a disjoint partition of all the categories in the ontology [5].

The second principle for ontology design is based on combinations. The combinations classify the objects into *firstness* versus *secondness* versus *thirdness*, or *Form* versus *Role* versus *Mediation*. These categories are not mutually exclusive. For example, *Woman* is considered to be a form (firstness) because it can be defined without considering anything outside a person. As a mother, a teacher, or an employee, the same individual would be an example of a role (secondness). These roles represent an individual in relation to another type (a child, a student, an employer). Marriage is a mediation (thirdness) category because it relates several (in this case, two) types together, lecture class of AI is mediation as it relates many students as a type together.

The third principle is based on constraints. It is, *continuant* versus *occurrent*,or *object versus Process*. Continuants are objects that retain their identity over some period of time; occurrents are processes "whose form is in the state of flux". For example, *Avalanche* is a process, and *Glacier* is an object. Note that this distinction depends on the time scale. On a grand time scale of centuries, Glacier is also a process. These distinctions are combined to generate new categories (Fig. 6.7). At a lower level, for example, Script (a computer program, or a baking recipe) is a form that represents sequences and is thus defined as *Abstract*, *Form*, *Process*. Also, *History* (an execution of a computer program), which is a proposition that describes a sequence of processes, is then Abstract, Form, or object [7].

In the John Sowa's ontology, top level category is for every possible combination of *distinctions*. Constraints are used at the lower levels to rule out categories that cannot exist, as well it avoids too many combinations of categories. For example, *logical constraints*, would rule out triangles of more than three sides, and *empirical constraints* would rule out birds that talk or think. Representation of constraints is in the form of axioms, inherited through the ontology hierarchy to the lower levels.

6.6 Reasoning Using Ontologies

We have discussed in the above that, ontologies are taxonomical structures of categories, subcategories, and members of concepts. As human, we identify the objects, whether physical or abstract, due to this classification. For example, we can distinguish an oranges from cricket balls, because in the first the surface property belong to softness, and spongy, though their size and shape are quite similar. The Structures, e.g., a University System, a building, a computer network, operating system, or say, even office chair, are decomposed into parts, so that we identify the object as a whole as well the relationship of the parts remain explicit in this representation. Hence, it is due to these associated categorizations, and sub-categorizations, along with restrictions we are able to identify the real world objects. The reasoning can be done using categories and objects, using physical decomposition of categories, using measuring criteria, or using object oriented analysis.

6.6.1 Categories and Objects

The taxonomical hierarchies are common in government, military, and other organizations. Classifying objects into categories is important for knowledge organization. Of course interactions take place at individuals levels but relations are created at the level of categories. For example, the sentence, "Basketball team has won", does not refer to a single ball and nor a single player, but a relationship among team members as a group.

The Category also helps in prediction of objects once they are classified. One refers the objects from perceptual inputs, infers category of membership from perceived properties of the objects, then uses category information to predict about object. For example, from its yellow colour, size, shape, we identify that an object is a 'Mango'.

In FOPL there are two choices for categories: *predicates* and *objects*. Following are the examples:

$basketball(b)$;
$member(b, basketballs)$;
$b \in basketballs$; (the object b is a member of category basketball)
$subset(basketballs, balls)$; (category is subclass of another category)
$basketballs \subset balls$.
$dogs \in domesticatedanimals$. (categories are members)

The category is defined by *members* and *subset* relation. We also note that categories organize the knowledge as well as they help to inherit. Thus, 'mango'$\in$ 'mangoes', and 'mangoes'$\subset$ 'fruits'. Thus, mango inherits the properties of 'taste'from fruits (fruits have 'tastes'). Similarly, we have categories: animals, birds, foods, institutions. A "set of institutions" $\subset$ "institutions", and MBM $\in$ "institutions". The subclass organize the categories into a *taxonomic* hierarchy.

Following are the examples of reasoning through taxonomies:

$(x \in basketballs) \Rightarrow round(x)$; (all members of a category have same property of roundedness)
$color(x, brown) \wedge round(x) \wedge dia(x) = 9" \wedge x \in balls \Rightarrow x \in basketballs$; (the member category can be recognized by some property)

Having represented in taxonomy, when a top element is identified having some property, those its subordinates can be easily identified, as they possess similar properties, unless there is an exception.

6.6.2 Physical Decomposition of Categories

An object can be part of another, for example, nose, eyes, hands, are part of body; steering is part of car, wheels are part of wheel-assembly, and wheel-assembly is part of car. This can be represented by relations of physical decompositions,

$partof(wheel, wheelassembly)$.
$partof(wheelassembly, car)$.
$partof(x, y) \wedge partof(y, z) \Rightarrow partof(x, z)$.

Note that taxonomies are *transitive relations*. The relation of reflexivity also holds on individual objects, e.g., $partof(x, x)$, as x has one part, and that part is itself. The categories of composite objects is defined by structural relations between parts and assemblies.

6.6.3 Measurements

The physical objects have height, weights, mass, length, and other physical measurements. To characterize a physical object, values need to be assigned to the objects in the form of their properties. Following are the examples:

$length(ipad) = inches(5)$ or $length(ipad) = centimeters(12.5)$.
$listprice(basketball) = rupees(500)$.
$height(x) > height(y) \Rightarrow taller(x, y)$.

6.6.4 Object-Oriented Analysis

The ontology development processes are similar to that of object-oriented design and analysis. In both, domain vocabulary is collected in the beginning, which is often out of domain's generic *nouns*, *verbs*, and *adjectives*. The result of object-oriented

analysis is in the form of a draft document for domain ontology and relevant to the application. However, analysts do not call the result as ontology. A designer in an object-oriented system defines the things like, objects, classes, interface functions, hierarchies, and the system behavior. But, an ontological engineer make use of intermediate representations such as graphs, tables, and semantic networks, to design hierarchies and other concept relationships, which hold true for object oriented design also.

In object oriented analysis, the classes act as templates. Both types of specialists make use of templates to specify product details. The classes are then merged or refined, as with ontologies.

6.7 Ontological Engineering

The *knowledge engineering* is process of knowledge representation—identification of task, assemble relevant knowledge, decide vocabulary of predicates, functions and constructs, encode knowledge about domain, encode description of specific problem instance, pose queries to procedures and get answers and debug knowledge base.

The ontological engineering can be defined on the same line of knowledge engineering. It is related to the creating representation of general concepts—actions, time, physical objects and beliefs.

Ontological engineering comprise a set of activities conducted during the time of: conceptualization, design, implementation and deployment of ontologies. Ontological engineering covers topics of philosophy, metaphysics, knowledge representation formalisms, development methodology, knowledge sharing and reuse, knowledge management, business process modeling, common-sense knowledge, systematization of domain knowledge, information retrieval from the Internet, standardization, and evaluation. It also gives us design rationale of a knowledge base, helps define the essential concepts of the world of interest, allows for a more disciplined design of a knowledge base, and enables knowledge accumulation. In practice, knowledge of above mentioned disciplines helps to:

- Organize the knowledge acquisition process;
- Specify the ontology's primary objective, purpose, granularity, and scope; and
- Build its initial vocabulary and organize taxonomy in an informal or semi-formal way, possibly using an intermediate representation.

Special-purpose languages/tools used for implementing ontologies, such as *Ontolingua*, *CycL*, and *LOOM* (lexical OWL ontology matcher), use a frame-based formalism, a logic-based formalism, or combination. Ontolingua is a frame-based language that uses KIF (Knowledge Interchange Format). It is a language for publication and knowledge communication, with notation and semantics like some extended form of first-order predicate logic. Ontolingua enables writing knowledge-level specifications independent of particular data or programming languages, and can translate a knowledge base from one representation language into another.

The LOOM takes two ontologies represented in OWL and produces a pair of relatedconcepts from the ontologies. In order to identify the corresponding concepts, LOOM compares the preferred names and symbols of the concepts in both ontologies. It identifies these concepts as similar, if and only if, their preferred names or synonyms are equivalent based on modified string functions. The string comparison removes the delimiters from both strings, then uses approximate matching techniques to compare them, allowing for mismatch of at most one character in a specified length.

Some languages/tools used in building ontologies are based on translation approaches. Using these approaches it is possible to build ontologies directly at knowledge level, and knowledge level is translated into the implementation level, thus eliminating the need to master the implementation languages [8].

6.8 Situation Calculus

The concept of *action* arises in at least two major subareas of artificial intelligence, (1) natural language processing and (2) problem solving. For the most part, the formalisms that have been suggested in each sub-area are independent of each other and difficult to compare. However, so far there does not exists a computational theory of actions, which is powerful enough to capture the range of the meanings and distinctions expressible in any natural language. The primary goal of situation calculus is to provide a formalism which is expressive enough for representation of actions and to explore its use in defining the meanings of English language sentences that describe actions and events [9].

The requirement on the formalism for representation of actions is that it should be a useful representation for *action reasoning* (i.e., problem solving). It has a very useful application, i.e., to describe, how this representation could be used for planning (of actions) or for plan recognition system. This is essential to the natural language processing as well, because the natural language understanding is aimed to ultimately result to problem-solving capability.

The situation calculus is also the language of choice for investigations of various technical problems that arise in theorizing about *actions* and their *effects*. It is being taken as a foundation for practical work in planning, control, simulation, database updates, agent programming and robotics. In parallel with these developments of its applications, there have emerged axiomatizations for the situation calculus, and explorations of some of their mathematical and computational properties [10].

6.8.1 Action, Situation, and Objects

The situation calculus is a logic formalism for representing and reasoning about dynamical domains. The concepts in the situation calculus are *situations*, *actions* and *fluents*. Number of objects are typically involved in the description of the world.

The situation calculus is based on a *sorted domain* with three sorts: actions, situations, and objects, where the objects include everything that is not an action or a situation. Hence, domain D is,

$$D = \langle A, S, O \rangle \tag{6.2}$$

where A is set of actions, S is set of situations, and O is set of objects, with variables of all sort can be used. The actions, situations, and objects are elements of the domain, but fluents are modeled either as predicates or as functions. The phrase, e.g., sorted domain of actions means, that set of actions are to be carried out in a certain (sorted), and similar meaning exists for sorted situations, and sorted fluents.

A situation is a kind of state (*ako*), however it is a result of chain of earlier situations. Actions are what make the dynamic world change from one situation to another when performed by agents. A fluent is a condition that can change over time. Fluents are situation-dependent functions used to describe the effects of actions. There are two kinds of them: *relational fluents* and *functional fluents*. The former have only two values: *True* or *False*, while the latter can take a range of values. For instance, one may have a relational fluent called *handempty* which is true in a situation if the robot's hand is not holding anything. We may need a relation like this in a robot domain. One may also have a functional fluent, e.g., *battery-level*, whose value in a situation is an integer of value between 0 and 100 denoting the total battery power remaining in one's notebook computer.

The set of actions form sort of domain. The action can also use variables, and they can be quantified, for example in a robot world, possible action terms would be $move(x, y)$ to model a robot moving to a new location (x, y), and $pickup(o)$ to model the robot picking up an object o, and so on.

6.8.2 Formalism

A formalism used in situation calculus is aimed to implement the commonsense reasoning, such that the information used by humans can be expressed in sentences in logical order, then it is made part of the database (knowledge base). As next step, a goal oriented program will consult these databases for the required facts to ultimately achieve the goal. The database stores facts about what are the effects for any given actions, for example, a set of facts and effects concerning to a robot which moves objects from one location to another location. The situation calculus is designed to derive the actions (effects) for a given set of facts, independent of any specific problem.

The major part of the database are facts about effect which resulted due to actions, e.g., action of a robotic arm. These consequences of actions in situation calculus are of general nature, and not bound to specific problem, hence they can be used for solution of variety of problems, without creating new databases for new instances of problems.

In the situation calculus, a dynamic world is modeled to progress through a series of situations as a result of various actions being performed within this world. A situation is a consequence of sequence of action's occurrences. The situation available before any actions are performed is generally denoted by S_0, called *initial situation*. A new situation resulting from the performance of an action is denoted using the function symbol *result* or *do*. This function symbol has a 'situation'and 'action'as arguments, and a new situation as a result due to performing the given action in the given situation. We shall make use of *do*, called binary function symbol expressed as,

$$do : action \times situation \rightarrow situation. \tag{6.3}$$

The intended interpretation is that $do(a, s)$ (or $result(a, s)$) denotes the successor situation resulting from performing action a in situation s. Accordingly, the basic formalism of the situation calculus is represented by:

$$s' = do(e, s) \tag{6.4}$$

which asserts that s' is the resulting situation when event e (action) occurs in situation s.

The basic ontology of situation calculus consists of:

1. situations, which corresponds to snapshots of universe or an instant of time, and,
2. actions or events, which change the world from one state to another.

It is a sorted language with sorting (order) for situation and actions.

The fluent "$iscarrying(o, s)$" can be used to indicate that the robot is carrying a particular object 'o'in a particular situation 's'. If the robot initially carries nothing, "$iscarrying(Ball, S_0)$" is false while, the fluent

$$iscarrying(Ball, do(pickup(Ball), S_0))$$

is true. The location of the robot can be modeled using a functional fluent location, which returns the location coordinates '(x, y)'of the robot in a particular situation.

To describe a dynamic domain using the situation calculus, one has to decide on the set of actions available for the agents to perform, and the set of fluents needed to describe the changes these actions will have on the world. For example, consider the blocks world where some blocks of equal size can be arranged into some set of towers on a table. The set of actions in this domain depends on what the imaginary agent can do. If we imagine an agent to be a robot-arm that can be directed to grasp any block that is on the top of a tower, and either add it to the top of another tower or put it down on the table to make a new tower, then we can have the following actions:

$stack(x, y)$—put block x on block y, provided that robot is holding x, and y's top is clear. Being action, it is read "stack x on y", and shall not be confused with predicate.

$unstack(x, y)$—pick up block x from block y, provided that robot's hand is empty, x is on y, and x has top clear.
$putdown(x)$—put block x down on the table, provided that robot is holding x.
$pickup(x)$—pick up block x from the table, provided the robot's hand is empty, x is on the table and top is clear.
$move(x, y)$—move block x to position y.

To describe the effects of these actions, we can use the following relational fluents:

$handempty$—True in a situation if the robot's hand is empty.
$holding(x)$—True in a situation if the robot's hand is holding the block x.
$on(x, y)$—True in a situation if block x is on block y.
$ontable(x)$—True in a situation if block x is on the table.
$clear(x)$—True in a situation if block x has top clear.

For action $stack(x, y)$ to be performed in a situation, the fluent,

- $holding(x)$ must be true, and
- $clear(y)$ must be true.

Also, after $stack(x, y)$ is performed and it results to a new situation, the fluent,

- $on(x, y)$ will be true,
- $handempty$ will be true,
- $holding(x)$ will be false, and
- $clear(y)$ will be false.

The action $stack(x, y)$, along with present value of fluents, and resulting action with new value of fluents can be formally expressed as an axiom,

$$\forall x \forall y[[((holding(x) \wedge clear(y)) \rightarrow (stack(x, y)] \rightarrow on(x, y) \wedge handempty \wedge \neg holding(x) \wedge \neg clear(y)).] \quad (6.5)$$

Now, for example, we can say that for action $stack(x, y)$ to be performed in a situation, $holding(x)$ and $clear(y)$ must be true, and that after $stack(x, y)$ is performed, in the resulting new situation, $on(x, y)$ and $handempty$ both will be True, and $holding(x)$ and $clear(y)$ will no longer be True.

We only need the action $move(x, y)$ to take place, to move block x to position y. This can happen if the agent (robot) in its world moves a block from a clear position to another clear position. The block x is clear to move means that x is on the table with top clear, or it is top most position in a stake of blocks. Accordingly, $move(x, y)$ action can be expressed by axiom,

$$\forall x \exists y[(clear(x) \wedge clear(y)) \rightarrow move(x, y)] \rightarrow on(x, y) \wedge clear(x)]. \quad (6.6)$$

For describing the effects of this action two fluents are sufficient: $clear(x)$, and $on(x, y)$. The action $move(x, y)$ can be performed only when the situation "$clear(x)$, $clear(y)$, $x \neq y$" is *True*. The consequence of this action is that x is no longer at the place where it was earlier, but instead at location y in the resulting new *situation*.

6.8.3 Formalizing the Notions of Context

The axiomatic system we discussed above is justified only in some specific context. With a little creative thought it is possible to construct a more general context in an axiomatic system. However, in such a generalized context the exact form of axioms will not hold. We can appreciate this argument if we think of human reasoning apparent in natural language, for example, consider axiomatizing of the adjective (concept) "on", to draw correct inferences from the information presented in a English language sentence, in a world of space-craft on a Mars mission (as context), where flight crew answers a query of other crew as,

"The book is on the table."

As critic may propose to suggest about the precise meaning of 'on', causing difficulties about what can be between the book and the table or about how much gravity there has to be in a spacecraft in order to use the word "on" and whether centrifugal force counts. If the space-raft is orbiting the planet Mars! Thus, we encounter a Socratic puzzles over the issue of what the concepts mean in complete generality and come across examples that never arise in reality—"there simply is not a most general context"!

On the other hand, if a system is axiomatized at sufficiently higher level (i.e., more general), the axioms will be too long to be suitable even in special situations. Thus, we find it convenient to say, "The book is on the table," and omit the reference to exact time when it is on the table (global or local), its precise location on the table (on which part or corner it is located), and what is location of table in reference to global (GPS) coordinates, or universal coordinates, etc. Hence, we conclude that position of book in above example is sufficiently general. How much general the specification be? This depends on whether that general/commonsense knowledge has been expressed in the logic used for reasoning, in the corresponding program or in other other formalism.

A possible solution to the question—"how much general?", is, formalize the idea of context, and combine it with circumscription procedure of *monotonic* reasoning. This is done by adding context parameters to predicates and functions in the axioms, because each axiom makes an assertion about certain context. Apart from this, the axioms should express that, facts are inherited using more restricted context unless exceptions are specified. Each assertion is assumed to apply nonmonotonically in any particular more general context, but for that also there are exceptions. For example, there is rule that says, birds fly. In this rule, we implicitly assume that there is an atmosphere available. However, in a more general context, this (existence of atmosphere) is not required to be assumed. Further, it still remains to be determined as how the inheritance to more general contexts differs from that of more specific contexts.

There are a some predicates about contexts as well as dealing with time. One of the most important predicate is $holds$, which asserts that a property holds (i.e., is true) during a time interval. Thus, the sentence,

$holds(p, t)$

is true if and only if property p holds during time t. A subsequent axiom will state, this is intended to mean that p holds at every subinterval of t as well. Note that if we had introduced *holds* as a "modal operator" we would not need to introduce properties into our ontology.

Suppose that, whenever a sentence p is present in the memory of a computer, we consider it as in a particular context and as an abbreviation for the sentence,

$holds(p, C)$

where C is the name of a context. We create a relation about the generality of a context as follows: a relation ($\leq$), e.g., $c_1 \leq c_2$ means context c_2 is more general than context c_1. Using this relation, we allow sentences like,

$holds(c_1 \leq c_2, C)$

which means that statements relating contexts ($c_1 \leq c_2$) can have contexts C.

A logical system using contexts might provide operations of *entering* and *leaving* a context yielding what we might call *unnatural deduction* allowing a sequence of reasoning as given in the followings.

$$
\begin{aligned}
&holds(p, C)\\
&ENTER\ C\\
&p\\
&...\\
&...\\
&q\\
&LEAVE\ C
\end{aligned}
$$

This resembles the usual logical natural deduction systems.

Example 6.1 Represent the following sentences in the formalism of situation calculus.

"A man called Rex is standing at the gate of Ghana Bird Sanctuary wearing a black T-shirt. Rex loads his toy gun, waits for few seconds, and shoots at a migratory crane."

The fluents, which are either true or false, in this sentence are:

standing(place): whether Rex is standing at a given place or not?
black: whether Rex is wearing black T-shirt or not?
loaded: whether the gun is loaded or not?

Following are the actions in above sentence:

$load$: Rex is loading the gun.
$wait$: Rex waits for few seconds.
$shoot$: Rex shoots his gun.

Now, let us find out what holds at initial situation S_0.

$holds(standing(gate), S_0)$
$holds(black, S_0)$
$holds(alive, S_0)$
$\neg holds(loaded, S_0)$

Now we try to relate actions with situations. In other words, which fluents will hold after performing a certain action in a given situation?

1. If Rex shoots the gun and gun is loaded, crane is not alive.
2. If Rex shoots the gun, gun will become unloaded.
3. If Rex loads the gun, gun will be loaded.
4. If Rex waits on an loaded gun, the gun remain loaded.
5. If Rex waits on an unloaded gun, the gun remain unloaded.

Our first method for writing above sentences in formal way is as follows. Consider a general sentence "f_2 (fluent) will be true by performing action a_2 in state s if (provided that) f_1 is true in s."

$$\forall s[holds(f_1, s) \rightarrow holds(f_2, do(a_2, s))]$$

Now consider the first sentence: "if Rex shoots the gun and the gun is loaded, crane is not alive," which is written as:

1. $\forall s[holds(loaded, s) \rightarrow \neg holds(alive, do(shoots, s))]$
2. The remaining four sentences can be expressed in situation calculus as follows:
3. $\forall s[\neg holds(loaded, do(shoot, s))]$
4. $\forall s[holds(loaded, do(load, s))]$
5. $\forall s[holds(loaded, s) \rightarrow holds(loaded, do(wait, s))]$
6. $\forall s[\neg holds(loaded, s) \rightarrow \neg holds(loaded, do(wait, s))]$

Having represented in this form of FOPL, the reasoning is done using conventional methods used for predicate logic reasoning, e.g., resolution based theorem proving.

6.9 Nonmonotonic Reasoning

The most logic we have studied so far falls in the category of *monotonic logic*, and the corresponding reasoning as *monotonic reasoning*. The property of *monotonicity* is satisfied by all methods that are based on the classical (mathematical) logic. As per this property, if a conclusion is warranted on the basis of certain premises, no

additional premises should ever invalidate the conclusion. The Nonmonotonic logic is those ways of reasoning to infer additional information, that do not satisfy the monotonicity property of classical logic, that is, on the face of additional information, some of the earlier conclusions drawn may even become invalid!

In everyday life, however, we often draw sensible conclusions from what we know/information available to us. On receiving new/updated information, many a times we take back our previous conclusions, even when the new information we have collected has in no way indicated that it is contradicting the previous conclusions. Consider the following example: we have assumption that most birds fly, and that penguin are the birds which do not fly. On learning or receiving a new information that "Tweety is a bird", we infer that it flies. Further learning that "Tweety is a penguin", still does not effect our old inference that most birds fly. But, it (addition of new information) will require to abandon on inference of "Tweety flies." It is quite appropriate to say that intelligent automated systems must have the capabilities of this kind of inference, call as non-monotonic inference.

Many systems perform such nonmonotonic inferences. The most common are : negation as failure, circumscription, the modal system, default logic, autoepistemic logic and inheritance systems. Each of those systems is worth studying by itself, but a general framework in which those many examples could be compared and classified is necessary. The following section presents a general framework, concentrating on properties, which are important families of nonmonotonic reasoning systems.

6.10 Default Reasoning

One of the features of commonsense reasoning that makes it different from traditional mathematical proofs is the use of *defaults*. A default is a proposition that is postulated to be true in the absence of information to the contrary. For instance, an intelligent agent may assume by default that his observation correctly reflects the state of the world, and be prepared to retract this assumption in the face of evidence that be is in error [11].

The logic systems we have often come across, like, *classical logic*, *intuitionistic logic*, and *modal logic* are monotonic in nature. The name monotonic indicates that on adding a new fact in these systems never results in retraction of a conclusion derived before addition of that new fact. In fact, after addition of new knowledge in these logic systems, the inference ability of these systems monotonically increases. The Default logic is called nonmonotonic, because the new fact may be an exception to one of the defaults included in the set, and on the face of that the inference system should withdraw the conclusions drawn already.

In an intelligent system (either computer-based or human), when it tries to solve a problem, it may rely on complete information about the problem, and its main task will be to draw correct conclusions using classical reasoning. The classical (predicate) logic may be sufficient in such cases. But, in many situations, the system may have only incomplete information at hand, and it is required to draw inferences

with only these available information. The nonavailability of the information may be because of the need to respond quickly, and acquiring these more information will take some time, but the system cannot afford to wait for the collecting of all the relevant data. For example, the case evacuation required in a nuclear disaster, where we cannot wait to gather the complete information, and then decide to evacuate or not, as otherwise it would have caused havoc. Some times, the information available is unreliable/inaccurate, and it needs to be authenticated, which will require more time, and we cannot prolong the decision.

The classical logic possess the quality to represent and reason with certain aspects of incomplete information. In special conditions, additional information needs to be "filled in" to overcome the incompleteness, as it is important to make certain decisions. In such special circumstances, the system need to make some plausible conjectures, which are based on *rules of thumb*, called *defaults*, and this modified logic is called *default logic*, and the reasoning is called *default reasoning*. As an example, a doctor has to make some conjectures during a situation of emergency, about some most probable causes of the symptoms observed. Obviously, in such conditions it would be not appropriate to wait for the results of laboratory tests, as possibly extensive and time-consuming tests results may arrive too late to complete the diagnosis based on those and begin the treatment.

However, when a medical diagnosis (i.e., decision) is based on assumptions, it may turn out to be wrong on the face of availability of new information, and may require a modified diagnosis! The phenomenon of having to take back some previous conclusions is called *nonmonotonic* reasoning or *non-monotonicity*, which can be stated as: if a statement φ follows from a set of premises M, i.e., $M \models \varphi$ and $M \subseteq M'$, then φ does not necessarily follow from M' (see Example 6.2). Default Logic provides formal methods to support this kind of reasoning.

Example 6.2 Let

$$M = \{\forall x(bird(x) \Rightarrow fly(x)), \forall y(penguin(y) \Rightarrow bird(y)), \\ \forall z(penguin(z) \Rightarrow \neg fly(z)), bird(tweety)\}.$$

Given these, we have $M \models fly(tweety)$, and $M' = M \cup \{penguin(tweety)\}$ is inconsistent. We would expect the inference

$$M \cup \{penguin(tweety)\} \models \neg fly(tweety), \tag{6.7}$$

making $fly(tweety)$ a defeasible consequent. □

The default Logic is the most commonly used method for nonmonotonic reasoning, due to the simplicity of the notion of default used in this, and because the defaults occur quite common in real-life situations.

6.10.1 Notion of a Default

A rule used by football organizers in Kashmir valley might be: "A football game shall take place, unless there is snow in the stadium." This rule of thumb is represented by the default

$$\frac{football : \neg snow}{takesPlace} \tag{6.8}$$

Consider the following example to understand the default reasoning: In the absence of information that there is going to be snowfall in the stadium, it is reasonable to assume $\neg snow$, and also to conclude that game will take place. Accordingly, the preparations for the game can go on. But actually if there is a heavy snowfall during the night before the game is scheduled, then this assumption is wrong. This is because, when we are certain, based on definite information that there is snow, we cannot assume $\neg snow$, and therefore the default cannot be applied. In this case we need to refrain from the previous conclusion (i.e., the game will take place), so the reasoning is nonmonotonic, as we are going to withdraw the previous conclusion [11].

Before proceeding with more examples, let us first explain why classical logic is not appropriate to model this situation. Of course, we could use the rule:

$$football \wedge \neg snow \rightarrow takesPlace. \tag{6.9}$$

The problem with this rule is that we have to definitively establish that there will be no snow in the stadium before applying the rule. But that would mean that no game could be scheduled in the winter, as it would not be possible to comment about "no snow" on previous night of game, for in advance to decide whether to proceed for preparations or not! And, if we wait for the previous night of match to make decision for match, then there is no time left to do preparations. It is necessary to analyze the difference between two statements: having "to know that it will not snow", and being able to "assume that it will not snow." The defaults works on the second, and it supports drawing conclusions based on assumptions.

The defaults can be used to model *prototypical* reasoning, i.e., the most instances of a concept carry some property. One example is the statement "Typically, children have (living) parents"; this statement can be expressed using the default logic as,

$$\frac{child(X) : hasParents(X)}{hasParents(X)} \tag{6.10}$$

A no-risk reasoning is another form of default logic based reasoning, which is concerned to situations where we need to draw a conclusion even if it is not the most probable one, as another decision may be more disastrous. One such example is the

commonly used principle of awarding justice in the courts of Law: "In the absence of evidence to the contrary, assume that the accused is innocent." In default form we write this as:

$$\frac{accused(X) : innocent(X)}{innocent(X)} \tag{6.11}$$

The interpretation of rule (6.11) is that if $accused(X)$ is known, and there is no evidence that $innocent(X)$ (at numerator) is false, then $innocent(X)$ (at denominator) can be inferred.

6.10.2 The Syntax of Default Logic

A *default theory* T is a pair $\langle W, D \rangle$, where W is set of first-order predicate logic formulas (called the facts/axioms or belief set of T) and a countable set D of default rules. A default rule δ is of the form

$$\delta = \frac{\varphi : \psi_1, \ldots, \psi_n}{\chi} \tag{6.12}$$

here $\varphi, \psi_1, \ldots, \psi_n, \chi$, are one or more *closed formulas* in predicate logic. The formula φ is called *prerequisite* of the inference rule (6.12), $\psi_1, \ldots, \psi_n$ are called *justifications*, and χ is *consequent* or *inference* of this rule δ.

It is important to note that the formulae in a default must be a *ground clause*. For example the formula,

$$\frac{bird(X) : flies(X)}{flies(X)} \tag{6.13}$$

is not a default according to the definition: it should have all clauses as ground clauses. Let us call this rule of inference as *open defaults* rule. An open default is interpreted as a default schema, and it may represents a set of defaults, which may be infinitely large in numbers.

A default schema is very much like a default, with the difference that $\varphi, \psi_1, \ldots, \psi_n, \chi$ in default are arbitrary predicate formulas (i.e., they may contain free variables). Where as, a default schema defines a set of defaults as,

$$\frac{\varphi\sigma : \psi_1\sigma, \ldots, \psi_n\sigma}{\chi\sigma} \tag{6.14}$$

for all *ground substitutions* σ that assign values to all free variables occurring in the schema. That means free variables are interpreted as being universally quantified over the whole default schema. Given a default schema

$$\frac{bird(X) : flies(X)}{flies(X)} \tag{6.15}$$

and the facts $bird(tweety)$ and $bird(sam)$, the *default theory* is represented as

$$\begin{aligned} T =& \langle W, D \rangle \\ =& \langle \{bird(tweety), bird(sam)\}, \\ & \left\{ \frac{bird(tweety) : flies(tweety)}{flies(tweety)}, \right. \\ & \left. \frac{bird(sam) : flies(sam)}{flies(sam)} \right\} \rangle. \end{aligned}$$

The Default Logic is a simple and commonly used method of knowledge representation and reasoning in real-world situations. It has following characteristics:

- The most important aspect is that it supports reasoning with incomplete information.
- Defaults inferences are found naturally in many applications, like in, medical diagnostics, reasoning related to legal issues, information retrieval, preparing specifications of systems, and as a logic in software.
- Default Logic is commonly used to model the reasoning with incomplete information, which was the original motivation, as well as a formalism that enables compact representation of information.
- Important prerequisites for the development of successful applications in these domains are,
  - basic concepts' understanding, and
  - there exists a powerful implementation of default logic.

6.10.3 Algorithm for Default Reasoning

The Algorithm 6.1 for default reasoning provides extensions to the formula set W. Let $T = \langle W, D \rangle$ be a closed default theory (unspecified variables are false) with a finite set of default rules D and formula W. Let P be the set of all permutations of elements of default set D. If $P = \{\}$, i.e., $D = \{\}$, or if W is inconsistent, then return default theory $Th(W)$ as the only extension of T. The set of justifications and beliefs are indicated by variables *JUST* and *BELIEF*, respectively.

Algorithm 6.1 Algorithm for Default Reasoning

1: P = All permutations of elements of D
2: **while** $P \neq \{\}$ **do**
3: Take a permutation, $perm = \{d_1, \ldots, d_n\} \in P$
4: $P = P - \{perm\}$
5: ;Initialization
6: $BELIEF = W, JUST = \{\}$
7: ;Application of defaults and consistency test
8: **for** $i = 1$ to n **do**
9: ;[assume $d_i = \frac{A_i : B_i}{C_i}$]
10: **if** $(BELIEF \vdash A_i) \wedge (BELIEF \nvdash \neg B_i)$ **then**
11: $BELIEF = BELIEF \cup \{C_i\}$
12: $JUST = JUST \cup \{B_i\}$
13: **if** $\exists A(A \in JUST$ and $BELIEF \vdash \neg A)$ **then**
14: exit the algorithm
15: **end if**
16: **end if**
17: **end for**
18: **end while**
19: Return Th(BELIEF)
20: End

Example 6.3 Find extensions of following default theory:

$$T = \langle W, D \rangle = \langle \{R(n) \wedge Q(n)\}, \left\{ \frac{R(x) : \neg P(x)}{\neg P(x)}, \frac{Q(x) : P(x)}{P(x)} \right\} \rangle$$

First consider the permutation of D as,

$$(d_1, d_2) = \left\{ \frac{R(x) : \neg P(x)}{\neg P(x)}, \frac{Q(x) : P(x)}{P(x)} \right\}.$$

At the begin, we initialize $BELIEF = \{R(n) \wedge Q(n)\}, JUST = \{\}$. As per Algorithm 6.1, $d_1 = \frac{A_1 : B_1}{C_1} = \frac{R(x) : \neg P(x)}{\neg P(x)}$. From algorithm, we note that $BELIEF \vdash R(n)$, and $BELIEF \nvdash \neg(\neg P(n))$. Thus, we add C_i, i.e., $\neg P(n)$ in the $BELIEF$. Also, add $\neg P(n)$ into $JUST$. We also note that, the justification does not negate the $BELIEF$, hence it is consistent.

Next, we repeat the loop of Algorithm 6.1, for $i = 2$: $(d_2) = \frac{A_2 : B_2}{C_2} = \frac{Q(x) : P(x)}{P(x)}$. Before this, the updated $BELIEF = \{R(n) \wedge Q(n), \neg P(n)\}$. We note that $BELIEF \vdash A_2$ (i.e., $Q(n)$), but $BELIEF \nvdash \neg B_2$ (i.e., $\neg P(n)$) does not hold. Also, $Q(n) \in JUST$ holds, but $BELIEF \vdash \neg Q(n)$ does not hold, so algorithm continues. This conclude that belief is stable (see Table 6.1).

When above is repeated for other permutation, i.e., (d_2, d_1), we have,

$$(d_2, d_1) = \left\{ \frac{Q(x) : P(x)}{P(x)}, \frac{R(x) : \neg P(x)}{\neg P(x)} \right\},$$

Table 6.1 Belief computation-1

Iteration	BELIEFS	JUSTIFICATIONS	Consistency test
Initial	$R(n) \wedge Q(n)$	{}	
$i = 1, d_1$	$\neg P(n)$	$\neg P(n)$	OK
$i = 2, d_2$	Stable		

Table 6.2 Belief computation-2

Iteration	BELIEFS	JUSTIFICATIONS	Consistency test
Initial	$R(n) \wedge Q(n)$	{}	
$i = 1, d_2$	$P(n)$	$P(n)$	OK
$i = 2, d_1$	Stable		

we get the extended belief set as shown in Table 6.2.

In conclusion, the T has two different extensions: $Th(\{R(n) \wedge Q(n), \neg P(n)\})$, and $Th(\{R(n) \wedge Q(n), P(n)\})$, but obviously, not both at th same time.

6.11 Summary

Ontology is a theory about the nature of being or the kinds of existent. It is a systematic arrangement of all the important categories of *objects* or *concepts* that exist in some field of discourse, such that the arrangement shows the relations between them (objects/concepts). When compared with First-Order Predicate Logic (FOPL), in the ontologies we are particular for knowledge *organization* as well as *contents*, where as in predicate logic, the emphasis is on knowledge contents only.

The language, logic, ontology are closely related to commonsense, reasoning, and type checking, respectively. The language, ontology, and logic are representations, where as common-sense, type-checking, reasoning, are respectively, the applications. The natural language 'understanding is reasoning'paradigm. In comparing various ontologies, they can be viewed at three different levels: *is-a* taxonomy of concepts, (2) *internal concept* structure and relation between concepts, and (3) the presence or absence of *explicit axioms*.

John Sowa stated his fundamental top-level principles for ontology design as "distinctions, combinations, and constraints". The *distinctions* is Physical versus Information, *combinations* classify the objects into firstness versus secondness versus thirdness, or Form versus Role versus Mediation. The *constraint* is, continuant versus occurrent, or object versus Process.

The *knowledge engineering* is process of knowledge representation comprising of: identification of task, assemble relevant knowledge, decide vocabulary of predicates, functions and constructs, encode knowledge about domain, and deploy it. The *ontological engineering* is aimed to create representation of general concepts—actions, time, physical objects and beliefs.

Some of the languages and tools for implementing ontologies are: *Ontolingua*, *CycL*, and *LOOM*, which use either a frame-based formalism, or a logic-based formalism, or both.

Classifying objects into categories is important for knowledge organization and reasoning. For example, the sentence, "Basketball team has won", does not refer to a single ball and nor a single player, but a relationship among team members as a group. The Category helps in reasoning, e.g., prediction of objects once they are classified.

The concept of *action* arises in two major subareas of artificial intelligence, (1) natural language processing and (2) problem solving. However, the formalisms that have been suggested in each sub-area are independent of each other and difficult to compare. The requirement on the formalism for representation of actions is that it should be a useful representation for *action reasoning* (i.e., problem solving). It has an important application to describe how this representation could be used for planning (of actions) or for plan recognition system.

A *situation* is a consequence of sequence of action's occurrences. We shall make use of *do* (a binary function symbol) expressed as,

$$do : action \times situation \rightarrow situation.$$

which results to a new situation given an action and a present situation.

Nonmonotonic logic is the study of those ways of inferring additional information from given information that do not satisfy the monotonicity property. The following systems perform nonmonotonic reasoning:

- Negation as failure,
- Circumscription,
- Modal logic system,
- Default logic,
- Autoepistemic logic, and
- Inheritance systems.

One important features of commonsense reasoning that makes it different from traditional classical reasoning is the use of *defaults*. A default is a proposition that is assumed to be true in the absence of information to the contrary, i.e., unless the contrary information is made available, it is taken as true. In addition, the default logic is the most commonly used method for nonmonotonic reasoning due to its simplicity and due to the notion of a default.

Exercises

1. Give the top level ontology of following structures, represent the concepts using relations, and attributes.
    a. University system—consisting of faculties, departments, teachers, classes, students, courses, etc.

   b. Organizational ontology of a manufacturing firm.
   c. Organizational ontology of a project based software company.
   d. Government system ontology with various bodies and their responsibilities.

2. Suggest some applications of Sowa's ontology.
3. Compare and contrast the Wordnet and Sowa's ontology, and explain the reasoning performed in both with small examples.
4. Represent the ontologies of the following worlds, and explain, how you will perform the question-answering using each of these ontology?

   a. Ontology of Shirt.
   b. Ontology of Dining table.
   c. Ontology of University system.

5. Suggest the approach, how you will generate e-learning exercises using the ontology.
6. What is unnatural deduction in reference to situation calculus? Give examples to justify your claims.
7. How the inheritance works in a world of contexts? For example, in space-craft, on earth, and when context changes from one to other?
8. Show some similarities between "contexts" and "properties" in expressing situation calculus axioms.
9. Consider a robotic-hand which can move between several bins, pickup an object from the bin if the hand is above the bin and the hand is empty. The hand can drop an object into a bin if the hand is holding an object and the hand is above the bin. Moving of hand from any bin to any other bin is always possible, it does not require any preconditions. The actions are:

   $drop(x, y)$ (drop object x into bin y)
   $move(y)$ (move hand to be above the bin y)
   $grab(x, y)$ (pickup object x from bin y).

   The fluents are:

   $holding(x, s)$ (the hand is holding x in situation s)
   $over(y, s)$ (the hand is over bin y in situation s)
   $in(x, y, s)$ (object x is in the bin y in a situation s).

   a. Write the axioms for $move$, $drop$ and $grab$ actions.
   b. Write the successor state axioms for all the $fluents$.

10. A robot is to pickup n ($n = 10$) cuboid lying on the table and drop one-by-one in a bucket, available nearby the table. Write the statements for sequential calculus. What are the *situations, actions*, and *fluents* here?
11. Consider the eight puzzle shown in Fig. 6.8 with initial and goal states (situations).
    The objective is to go from a initial situation to the goal situation. We are allowed to move a tile into the empty space if that tile is adjacent to the empty space (e.g.

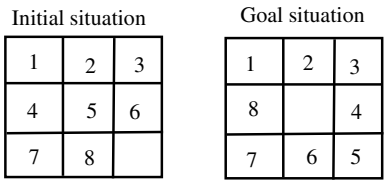

Fig. 6.8 Initial and final situation of 8-puzzle

in the initial situation tile 6 and 8 are adjacent to empty space). The locations are numbered as 1-9, as shown in initial situation, with number 9 as empty tile. The tiles are numbered 1-8. There is a single action $move(t, l)$ which indicates moving tile t to location l. Assume a predicate $adjacent(l_1, l_2)$, which is true when location l_2 is one move from l_1. It is only possible to do a move action if a tile is in a location adjacent to an empty location. The only fluent is $location(t, l, s)$ meaning tile t is in location l in situation s. Given this, write down:

- initial conditions,
- effect axioms,
- precondition axioms, and
- from the effect axioms derive the successor state axioms.

12. Given the following set of facts and default rules:

    a. People typically live in the same city where they work (default: d_1)
    b. People typically live in the same city where their spouses are (default: d_2)
    c. John works in New Delhi (fact: f_1)
    d. John's spouse works in Mumbai (fact: f_2)

    Answer these questions:

    a. Where does John live according to default logic?
    b. Where does John live according to your intuition?

13. Formalize these set of facts and default rules:

    a. Bob usually speaks the truth (d_1).
    b. John usually speaks the truth (d_2).
    c. Bob says that the suspect stabbed the victim to death (f_1).
    d. John says that the suspect shot the victim to death (f_2).
    e. Nobody can be both stabbed and shot to death (f_3).
    f. Stabbing or shooting to death is killing (f_4).

    Answer these questions:

    a. Did the suspect kill the victim according to default logic?
    b. Did the suspect kill the victim according to your intuitions?

14. Is the formula (6.10) sufficient and correct form of default reasoning, with X as variable? Justify your answer.

15. Translate the following into first-order predicate logic, and check whether the given conclusion follow from it?
    Typically, the computer science students like computers. Female students who like computers are typically interested in cognitive science. The computer science students are typically female: for example Anita, Babita, Cathy; but Dorthy is an exception to this rule. Conclusion: Anita, Babita, Cathy are interested in cognitive science; Dorthy is not interested in cognitive science.
16. Compute the default extensions of following theories $T = (M, D)$:
    a. $M = \{a\}, D = \{\frac{a:\neg b}{c}, \frac{:\neg c}{d}, \frac{:\neg d}{e}\}$
    b. $M = \{a \rightarrow c, b \rightarrow c\}, D = \{\frac{:\neg b}{a}, \frac{:\neg a}{b}, \frac{:\neg d}{e}\}$
    c. $M = \{\}, D = \{\frac{:\neg b}{a}, \frac{:\neg a}{b}, \frac{:\neg d}{d}\}$
    d. $M = \{p \wedge q\}, D = \{\frac{b:a}{a}, \frac{:\neg a}{a}, \frac{:\neg a}{\neg c}, \frac{:\neg q}{b}, \frac{:\neg p}{q}\}$
17. Compute the default extensions of $T = \langle M, D\rangle$, where

$$M = \{\forall x[mynah(x) \rightarrow \neg nests(x)],$$
$$\forall x[penguin(x) \rightarrow \neg flies(x)],$$
$$\forall x[birds(x) \equiv mynah(x) \vee penguin(x) \vee canary(x)],$$
$$bird(Tweety)\},$$
$$D = \{\frac{bird(x) : nests(x)}{nest(x)}, \frac{bird(x) : flies(x)}{flies(x)}\}$$

18. Find the extensions of the following default theories:
    a. $T = \langle\{\}, \{\frac{:\neg p}{p}, \frac{p \vee q:\neg p}{\neg p}\}\rangle$
    b. $T = \langle\{\neg Sun\text{-}shining \wedge Summer\}, \{\frac{Summer:\neg Rain}{Sun\text{-}shining}\}\rangle$
    c. $T = \langle\{\}, \{\frac{r:\exists x\ P(x)}{\exists x\ p(x)}, \frac{:r \wedge \neg p(x)}{r \wedge \neg p(x)}\}\rangle$
    d. $T = \langle\{p \vee q\}, \{\frac{:\neg p}{p}, \frac{p \vee q:\neg p}{\neg p}\}\rangle$
19. Assume that $\langle D, W\rangle$ be a propositional default theory, and D' be a set of normal defaults such that $D \subseteq D'$. If E is an extension of $\langle D, W\rangle$, then show that there exists an extension E' of $\langle D', W\rangle$ such that $E \subseteq E'$.

References

1. Chowdhary KR (2004) Natural language processing for word-sense disambiguation and information extraction. PhD thesis, Department of Computer Science and Engineering, J.N.V. University, Jodhpur (India)
2. Davis E, Marcus G (2015) Commonsense reasoning and commonsense knowledge in artificial intelligence. Commun ACM 58(9):92–103
3. https://protege.stanford.edu/ . Cited 19 Dec 2017
4. Douglas BL (1995) CYC: a large-scale investment in knowledge infrastructure. Commun ACM 38(11):33–38

5. Fridman-Noy N, Hafner CD (1997) The state of the art in ontology design: a survey and comparative review. AI Mag 53–74
6. http://clarity.princeton.edu/pub/wordnet/ . Cited 19 Dec 2017
7. Sowa JF (1995) Distinctions, combinations, and constraints. In: Proceedings of the workshop on basic ontological issues in knowledge sharing. Montreal, Canada
8. Devedzic V (2002) Understanding ontological engineering. Commun ACM 45(4):136–144
9. Pinto JA (1994) Temporal reasoning in the situation calculus. PhD Dissertation, Submitted to the Graduate department of Computer Science, University of Toronto
10. Thielscher M (1999) From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. Artif Intell 111:277–299
11. Antoniou G (1999) A tutorial on default logics. ACM Comput Surv 31(3)