

Ontological Processing of Sound Resources

LAC 2006
April 30, 2006

Jürgen Reuter
<http://www.ipd.uka.de/~reuter/>

Composers' Real Hard Life

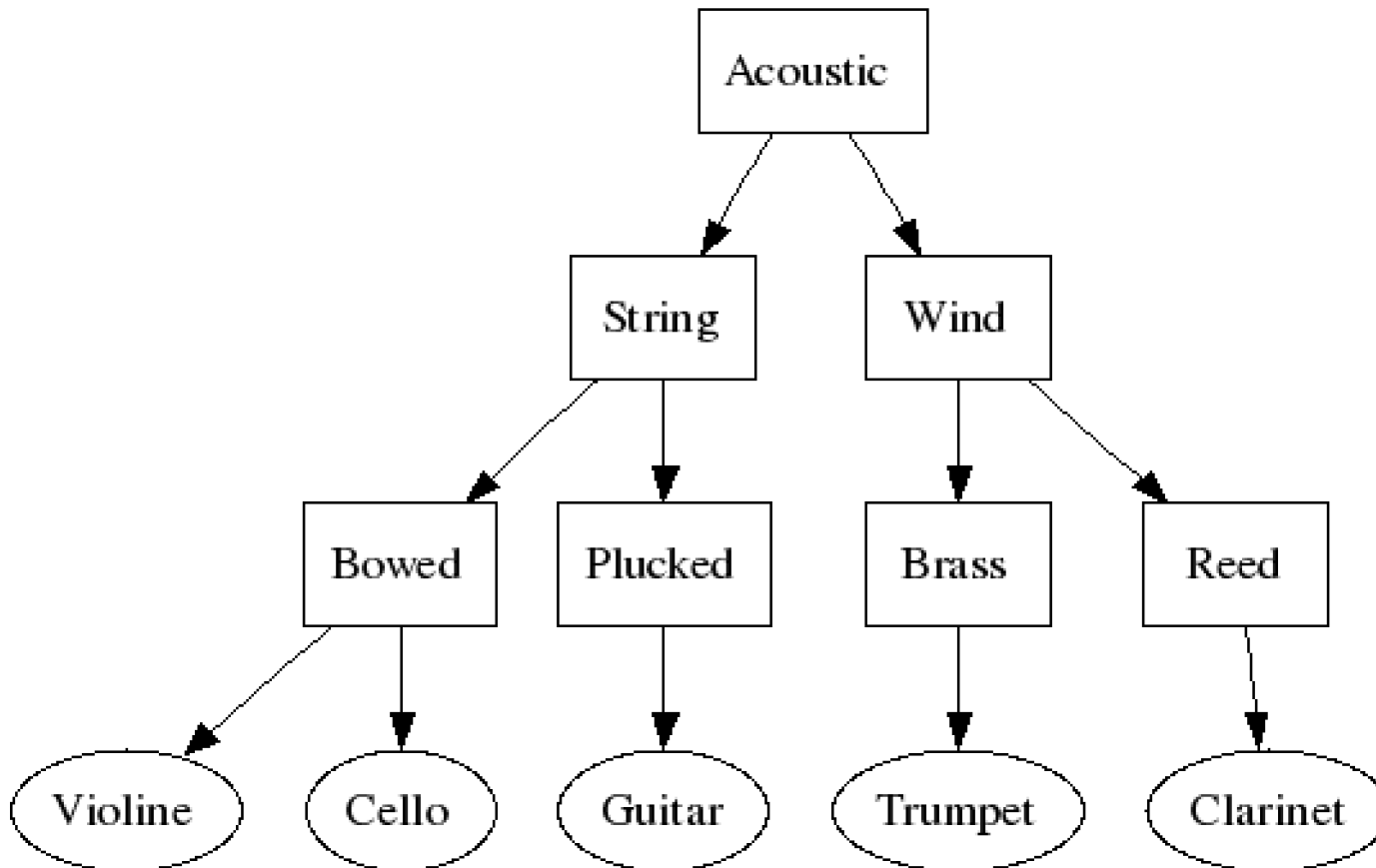
- “On **which synth** and in **what sound bank** was that cool trumpet sound?”
- “I somewhere saved that funny synth pad patch that I created for my last song, but **where did I store it?**”
- “Give me a **list** of all the **string synth sounds** that are scattered across my synths and banks!”

What's the problem?

- Too many sounds in too many synths and synth banks
- Sounds mostly not at all sorted or ordered in a musical sense
- No standardized, uniform way of sound browsing or lookup across synths
- No central sound registry for a single lookup of sounds across all synths in a system

How to Order Sounds?

- Instrument Taxonomies



How to Order Sounds? (cont.)

- Acoustic Organ Registers
 - Classify by **pitch**
 - 16", 8", 4", ...
 - Classify by **construction principle** of pipes
 - labial / lingual pipes, open / closed pipes, ...
 - Classify by **function** of sound
 - solo, principal, mixture, ...
 - Classify by similarity to **prototype sounds**
 - flute, bassoon, trumpet, ...

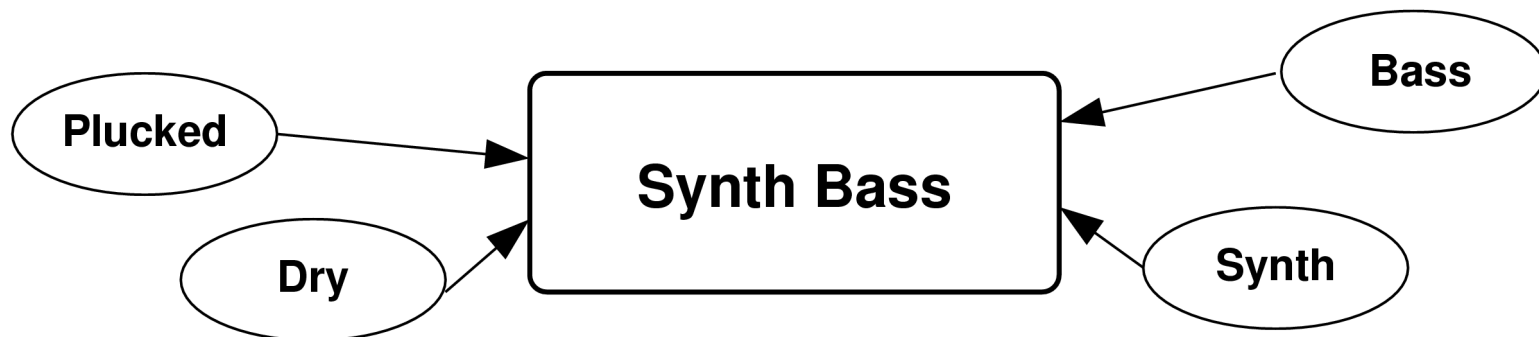
How to Order Sounds? (cont.)

- Grouping, banking
 - 128 GM Level 1 MIDI instruments, 16 groups
 - GM Level 2 banks

	<i>Piano</i>	<i>Chr.Perc.</i>	<i>Organ</i>	<i>Guitar</i>	<i>Bass</i>	<i>Strings</i>	<i>...</i>	<i>EFX</i>	
<i>Resonant</i>									Bank #20
<i>Bright</i>									Bank #16
<i>Slow</i>									Bank #8
<i>GM</i>									Bank #0
	Group #1	Group #2	Group #3	Group #4	Group #5	Group #6	<i>...</i>	Group #16	

How to Order Sounds? (cont.)

- Tagging
 - Generalizes grouping
 - Enables sound to be member of multiple groups
 - Serves for annotating qualities of a sound



More on Tagging

- Find (at least) 4 types of tags
 - **prototype-driven** (similarity to known sound or group of sounds)
 - string, violin, synth, percussive, bright, resonant, ...
 - **function-driven** (purpose of sound)
 - effect, lead, melodic, drums, ...
 - **construction-driven** (way of creating)
 - arpeggiator, decay, FM, vocoder, ...
 - **user-defined**
 - favorites, ...

More on Tagging (cont.)

- Tags are deductive
 - violin \Rightarrow string
 - drum \Rightarrow percussive
 - vocoder \Rightarrow synth
 - lead \Rightarrow melodic
 - ...

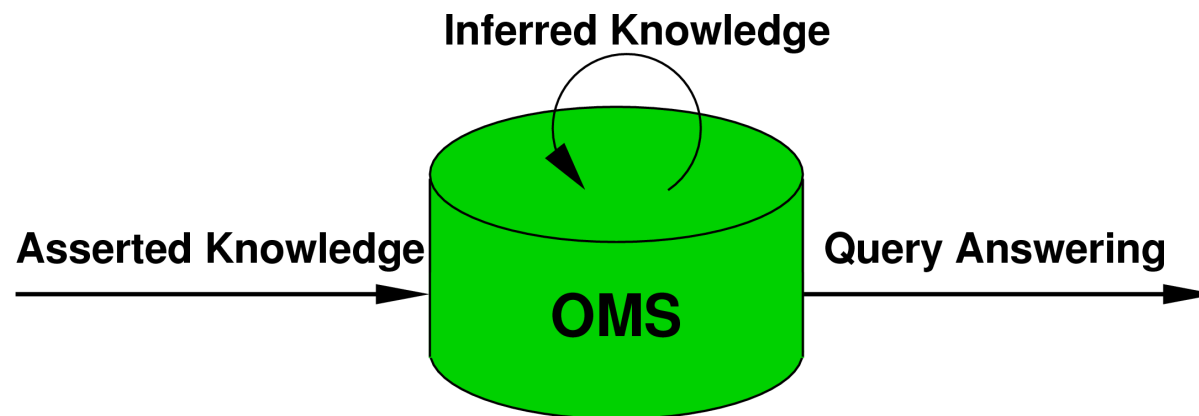
\Rightarrow plain Relational Database Management System (RDBMS) not sufficiently expressive

Ontology Management System (OMS)

- Builds upon description logics (aka concept languages)
 - represents decidable fragment of first-order logic
 - supports modeling in terms of classes, properties, and individuals
- Recently has become widely supported through OWL standard

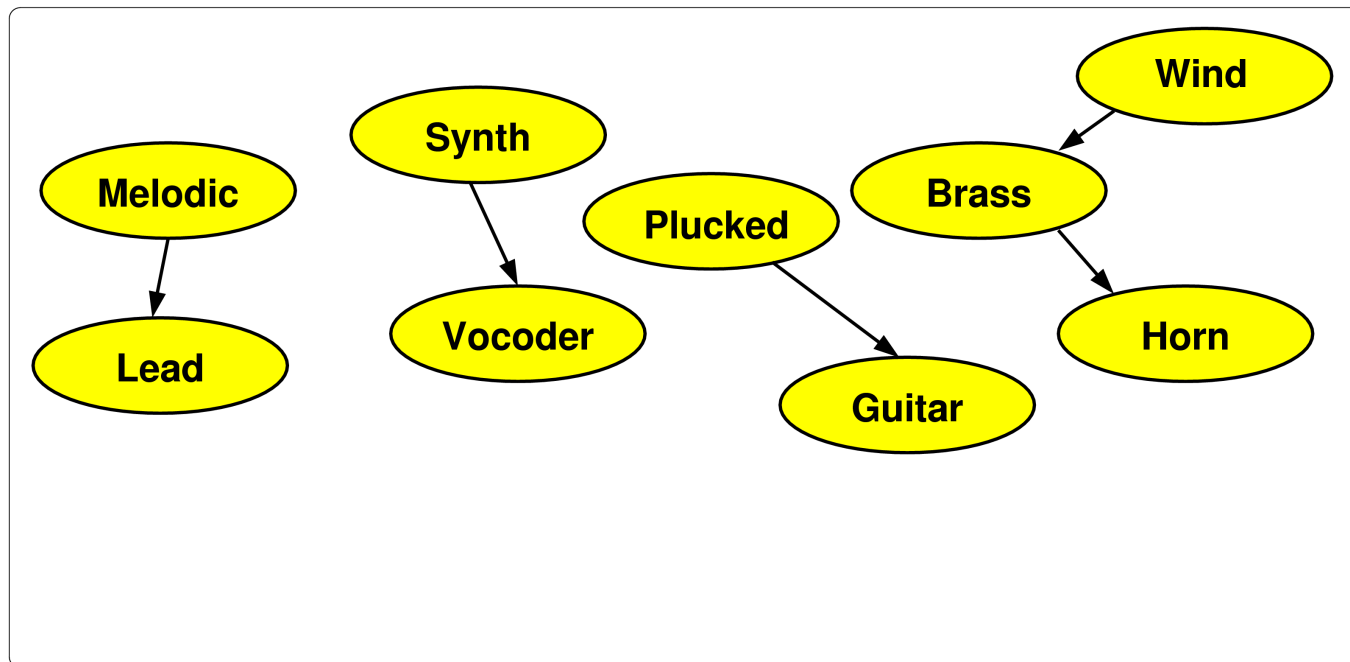
OMS vs. RDBMS

- Like a RDBMS, can serve as a central repository of information
- Unlike a RDBMS, also provides reasoning support for deducing knowledge



Ontologies

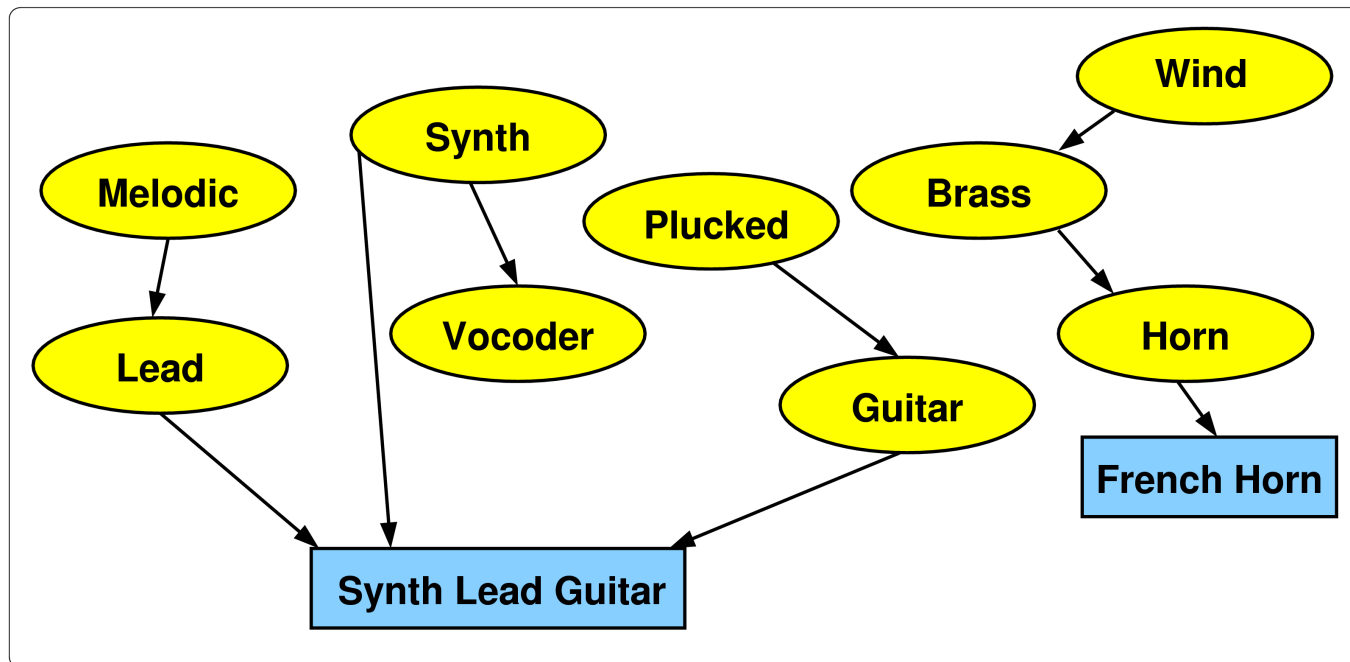
- **Classes**
 - create classes for tags and groups of sounds



Ontologies (cont.)

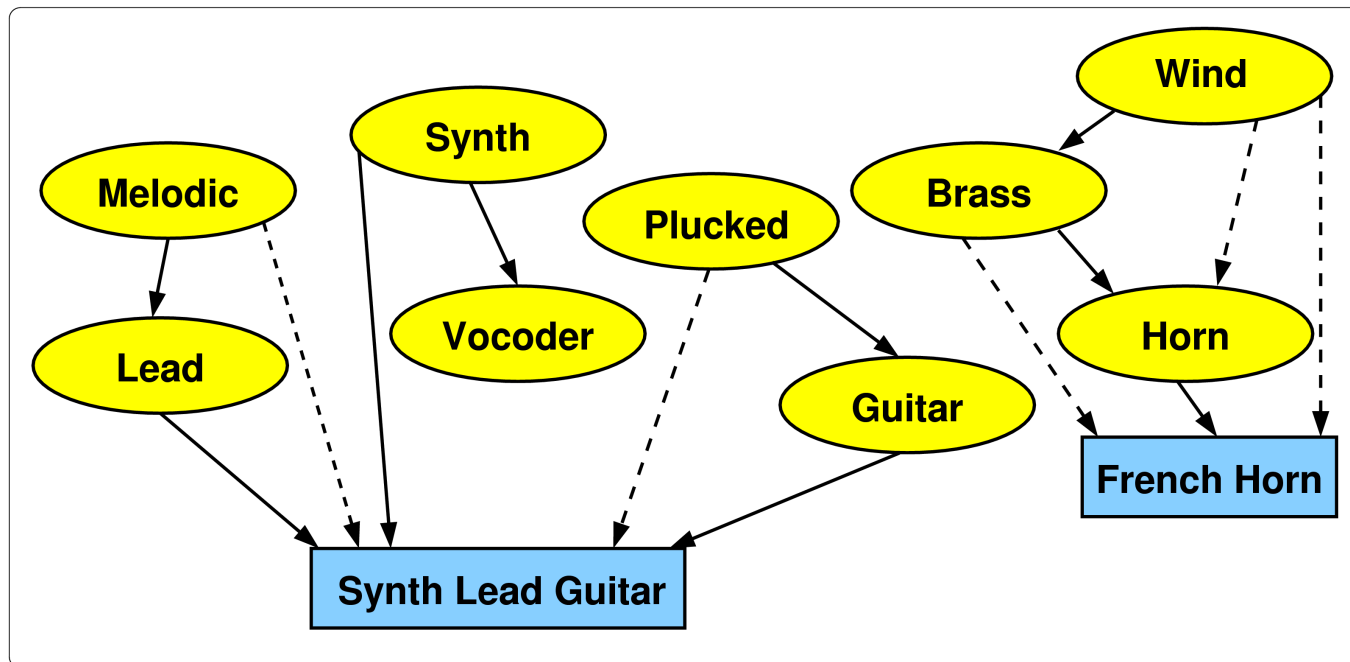
- Individuals

- store actually available sound resources as class members



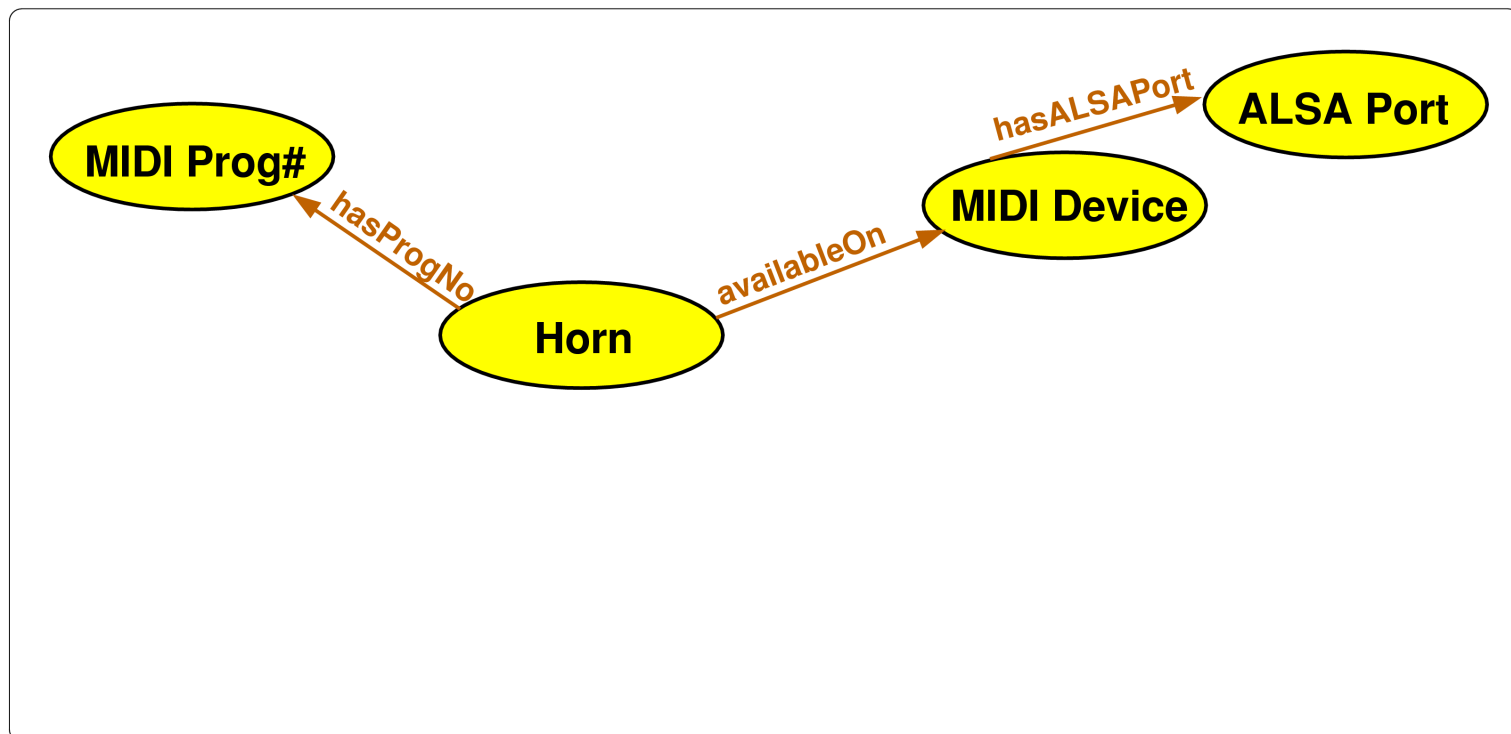
Ontologies (cont.)

- Individuals
 - infer inherited class memberships



Ontologies (cont.)

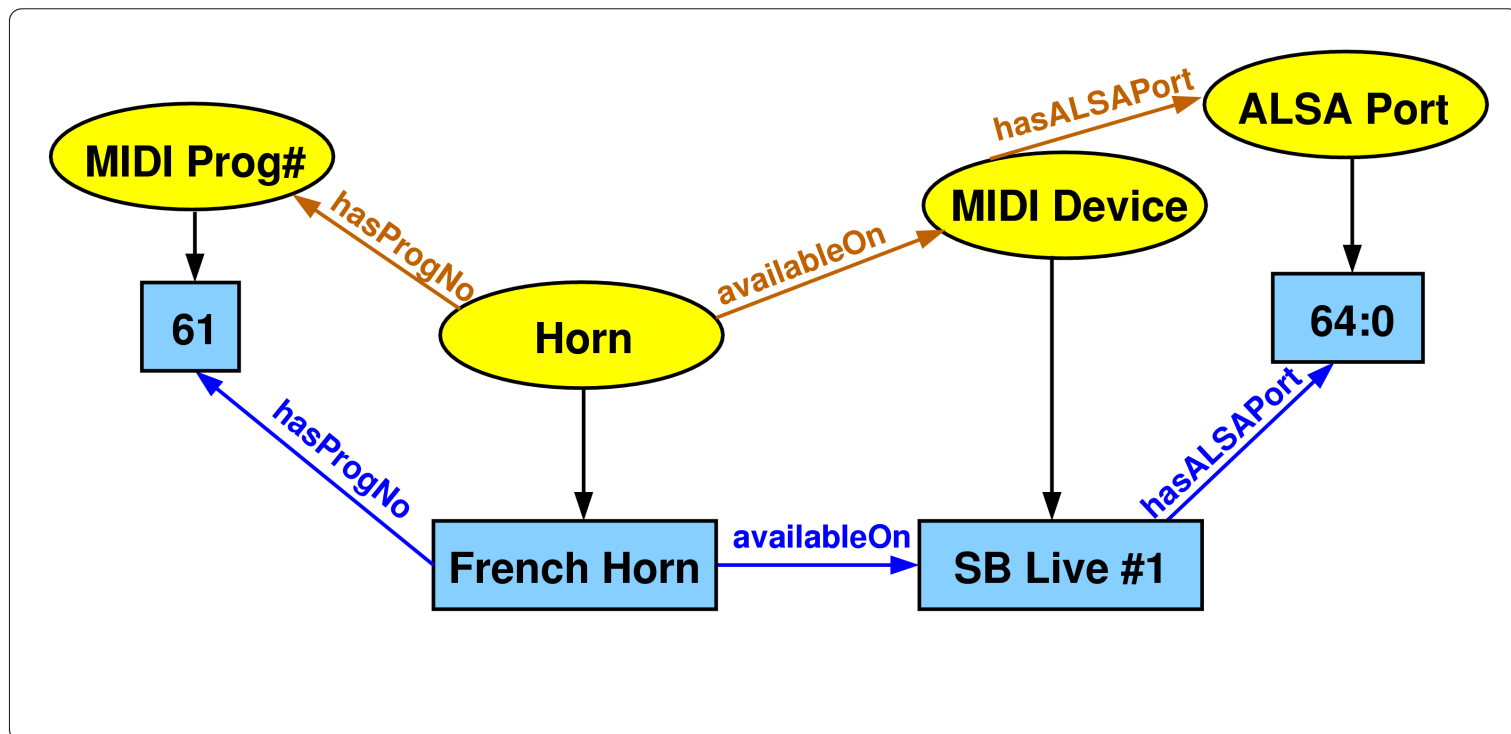
- Properties
 - associate each sound resource with related info (e.g. MIDI program number, ALSA port)



Ontologies (cont.)

- Properties

- associate each sound resource with related info (e.g. MIDI program number, ALSA port)

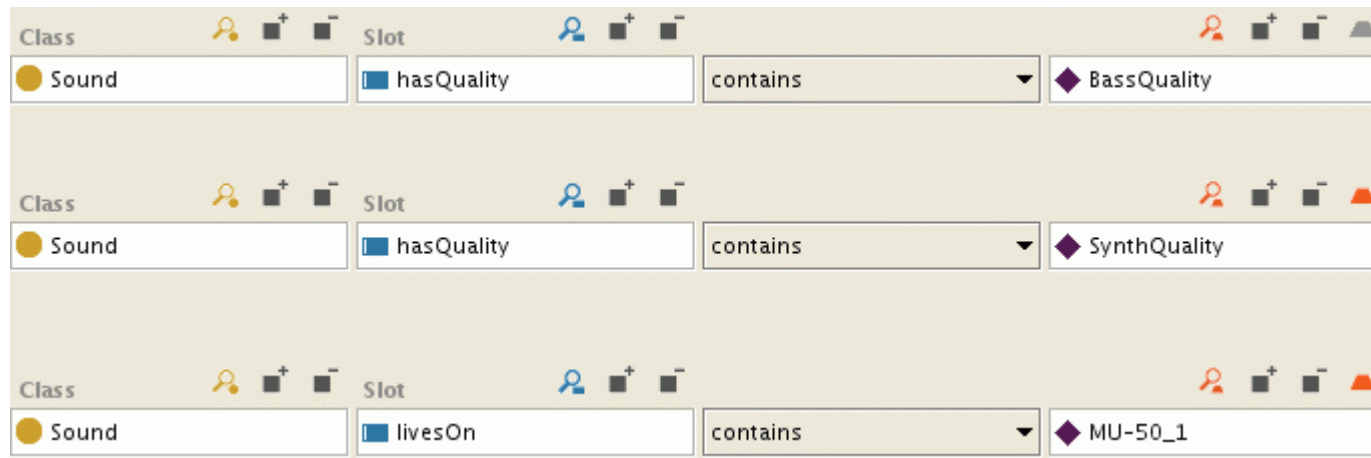


Example: Protégé

- Ontology editor and knowledge-base framework
 - Developed at Stanford University
 - Open source (Mozilla Public License)
 - Supports ontology editing, browsing, consistency checking, reasoning, ...
 - Used here to demonstrate feasibility of ontological sound resource processing

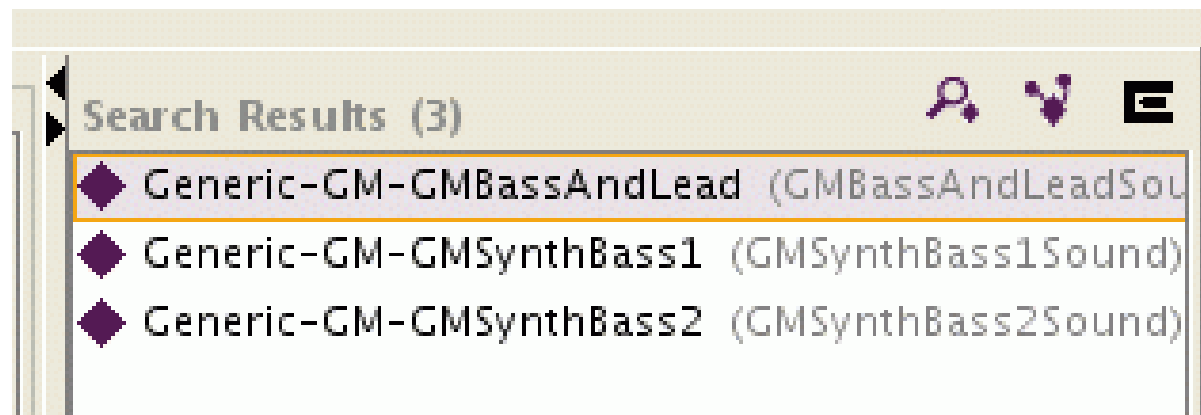
Example: Protégé (cont.)

- Query: $\text{Sound} \sqcap$
 $\exists \text{hasQuality}.\text{BassQuality} \sqcap$
 $\exists \text{hasQuality}.\text{SynthQuality} \sqcap$
 $\exists \text{livesOn}.\text{MU-50_1}$



Example: Protégé (cont.)

- Search yields 3 matches:



Example: Protégé (cont.)

- Result details

The screenshot displays the Protégé interface for an individual instance named "Generic-GM-GMBassAndLead" (instance of GMBassAndLeadSound). The interface is divided into several sections:

- Name:** "Generic-GM-GMBassAndLead"
- Annotations:** A table with columns "Property", "Value", and "Lang". It contains two entries:

Property	Value	Lang
rdfs:comment	A generic instar...en	en
rdfs:label	Generic GM Sound '...en	en
- Textual Description:** "A generic instance of a Sound that represents a 'Bass And Lead' Sound (program no. 88) on any GM Level 1 compatible SoundResource."
- has Name:** "Bass And Lead" (type: string)
- has Voices Left:** "16" (type: nonNegativeInteger)
- maps ToMIDIProgram:** "88" (type: nonNegativeInteger)
- has Quality:** A list of three quality types: "BassQuality", "LeadQuality", and "SynthQuality". This list is circled in red.
- lives On:** "MU-50_1" (type: GMBassAndLeadSound). This entry is also circled in red.

Example: Protégé (cont.)

- More on the “Lead” sound quality

The screenshot shows the Protégé Individual Editor for the instance 'LeadQuality' (instance of SoundQuality). The window title is 'LeadQuality (instance of SoundQuality)'. The main area is titled 'INDIVIDUAL EDITOR' and shows the instance name 'LeadQuality' (instance of SoundQuality). Below this, there are tabs for 'Name', 'SameAs', and 'DifferentFrom'. The 'Name' tab is active, showing the name 'LeadQuality' in a text box. Below the text box, there is a search icon and a text area containing the description: 'A Sound is said to have lead quality if it is useful rather for contributing to the melody of a music piece rather than to its harmonics or rhythm.' To the right of the text area is a vertical scrollbar. Below the text area are three icons: a download icon, a green bug icon, and a yellow star icon. On the right side of the editor, there is an 'Annotations' section with a table showing the following data:

Property	Value	Lang
rdfs:comment	A Sound is said ...	en
rdfs:label	Lead Quality	en

@ ALSA Developers

- Based on this presentation,
 - further elaborate a proper ontology
 - set up OMS that
 - serves as central sound registry
 - tracks available sound resources
 - design a registry management API
- Maybe promote ontological framework as cross-platform standard

@ Synth Application Developers

- Announce sounds to central registry
- Annotate sounds with tags
- Announce tags to central registry
- Think about proper tags for standardization
 - Can lead to much cleaner synth design!
- Let synth GUI design be guided by ontology of tags

Conclusion

- Management of sound resources is strongly desired.
- It is feasible based on an ontological framework.
- An OMS can serve as central registry.
- Applications should use query and update the OMS database.

Questions?