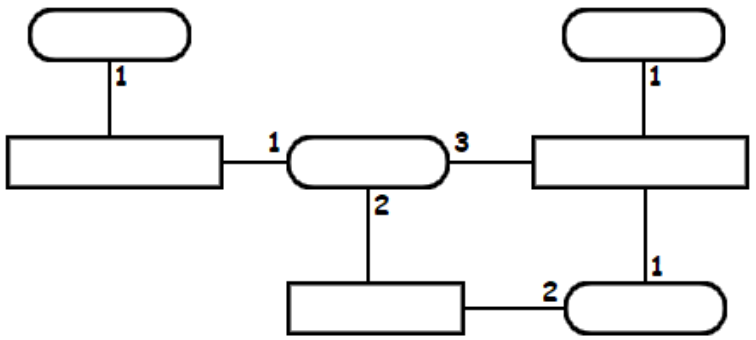
ERic v0.3 & Conceptual graphs

1. Abstract hyper-graphs

An abstract hyper-graph G(V,E) consists of a set V of vertices and a multiset E of hyper-edges.
Every hyper-edge in E is a tuple $(v_1,v_2,...,v_N)$ of vertices from the set V.
Thus an hyper-edge can connect many (1 to n) vertices.

There are graphical conventions to present an abstract hyper-graph. A vertex is drawn as an empty square box whereas an hyper-edge is drawn as an empty rounded box with each connection labeled with number 1 to n.

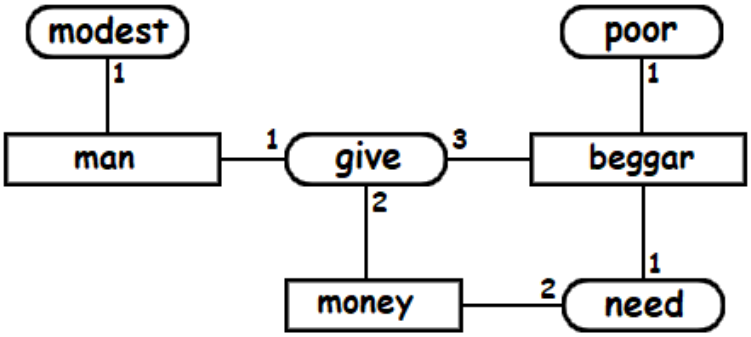
Example.



2. Concrete hyper-graphs

A concrete hyper-graph $G(V,E,L_V,L_E)$ is an abstract hyper-graph G(V,E) where each vertex is labeled with an element of L_V and each hyper-edge is labeled with an element of L_E.

Example.
A modest man gives some money to a poor beggar in need.

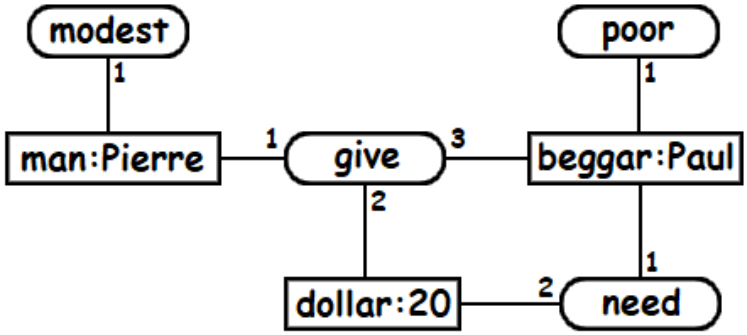


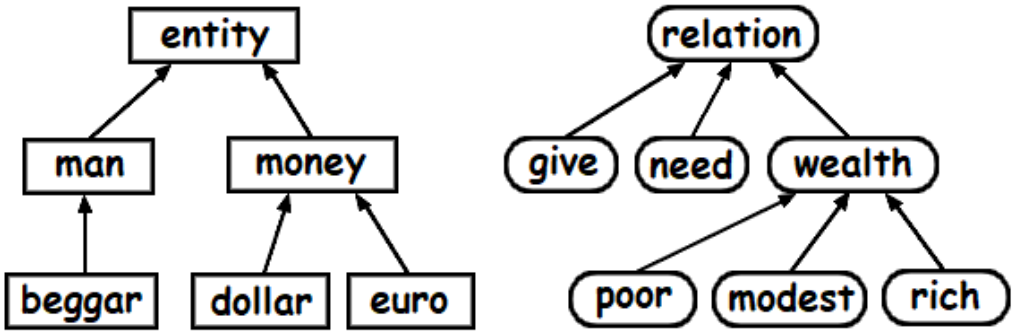
3. Conceptual graphs

A conceptual-graph $G(V,E,H_V,L_V,H_E)$ is an abstract hyper-graph $G(V,E)$ where :

- Each hyper-edge is labeled with an element of H_E where H_E is a hierarchy of hyper-edge labels.
- Each vertex is labeled with an element of H_V where H_V is a hierarchy of vertex labels and eventually also labeled with an element of L_V where L_V is a set of vertex labels.

Example : Pierre, a modest man, gives 20 dollars to Paul, a poor beggar in need.





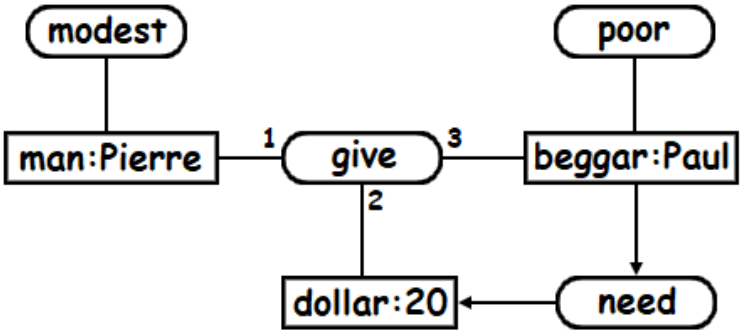
`entity` is the top-most concept of the H_V hierarchy.
`relation` is the top-most relation of the H_E hierarchy.
`Pierre`, `Paul`, `20` are elements of the set L_V.

4. More graphical conventions

In order to streamline the graphical notation :

- an edge connected to only 1 vertex is exempted from the 1 label.
- an edge connected to 2 vertices will have an input arrow & an output arrow instead of 1 & 2 labels.
- thus only ternary (and more) hyper-edges will retain connection labels.

Example : Pierre, a modest man, gives 20 dollars to Paul, a poor beggar in need.



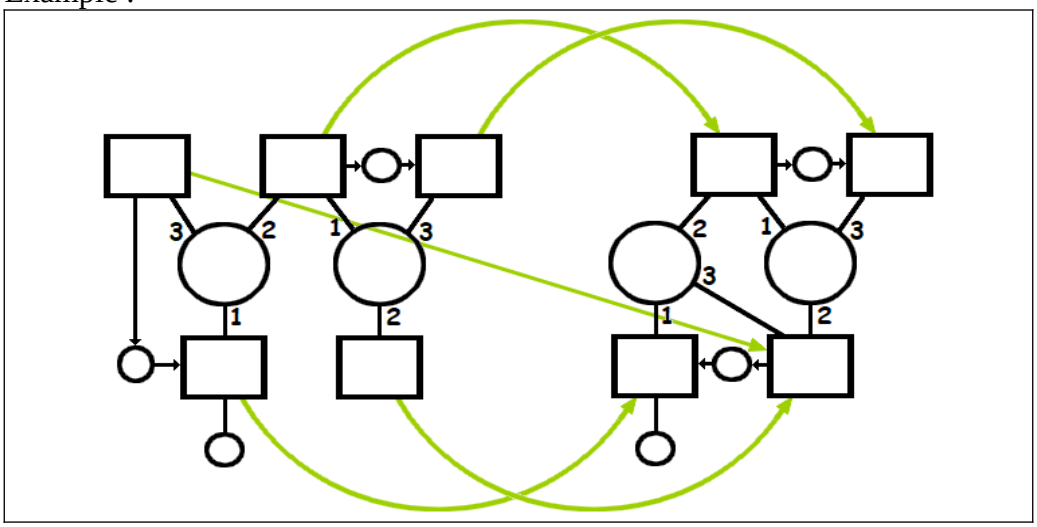
5. Abstract hyper-graph homomorphisms

An abstract hyper-graph homomorphism H from an abstract hyper-graph G(V,E) to an abstract hyper-graph G′(V′,E′) is a mapping *h* from V to V′ such that :

• informally : *h* preserves the adjacency relation.

• formally : if $(v_1,v_2,...,v_N) \in E$ then $(h(v_1),h(v_2),...,h(v_N)) \in E'$.

Example :



The abstract-hyper-graphs G(V,E) and G′(V′,E′).
The green arrows is a mapping from V to V′.

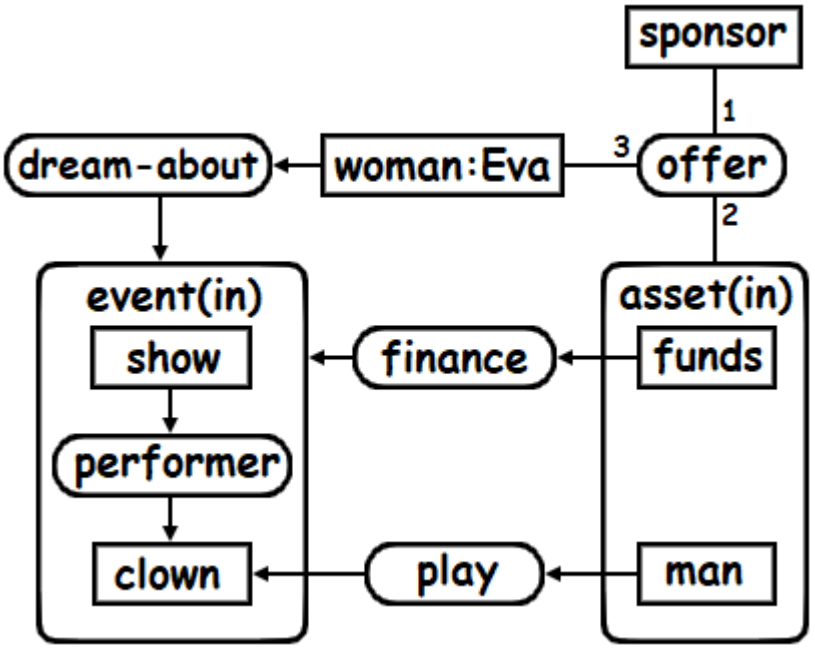
6. Conceptual graph homomorphisms and subsumption

The concept / relation hierarchies explicit an `is-a` relation between lower concepts / relations and higher concepts / relations. Similarly there is also an `is-a` relation between an entity and its concept. And finally there is an `is-a` relation between an abstract hyper-graph G′ and another hyper-graph G that maps to G′ by an homomorphism. When the three are taken together there is an `is-a` relation between G′ and G. Then it is said that the conceptual-graph G subsumes the conceptual-graph G′.

7. Nested conceptual graphs

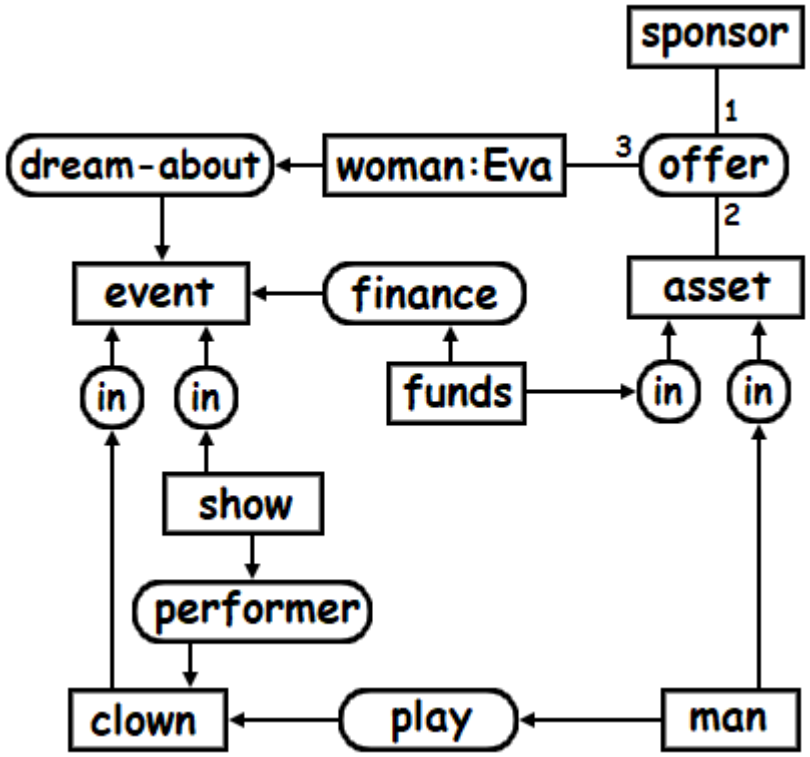
A nested conceptual graph is a conceptual graph where one (or more) square box contains further conceptual graph(s). Eventually, nested conceptual graphs allow edges to traverse square box boundaries in any manner (from inside to outside, from outside to inside, whatever the nesting level). Nested graphs provide a direct visual way to decorate the information with the associated context.

Example. Eva dreams about a clown show. A sponsor makes her an offer of a man to act as the clown and the funds to finance the event.



A nested conceptual graph can be unfolded, that is no box is nested, instead inner boxes are connected by a relation to outer boxes. In our example nesting is a convenient way to distribute the `(in)` relation.

Unfolded example.
Eva dreams about a clown show. A sponsor makes her an offer of a man to act as the clown and the funds to finance the event.



8. Nested conceptual graphs and subsumption

Theorem : there exists a bijection between unfolded conceptual graphs and nested conceptual graphs. As a consequence the subsumption between nested conceptual graphs is the same as subsumption between unfolded conceptual graphs.

9. CGIF (Conceptual Graph Interchange Format) notation

CGIF is a normalized textual notation for conceptual graphs communication. ERic v0.3 uses a CGIF dialect instead of the more appealing visual notation.

As a first demonstration our first conceptual graph translates to the following verbatim :

```
insert
  [man:Pierre]
  [beggar:Paul]
  (modest Pierre)
  (poor Paul)
  (give Pierre [dollar:20] Paul)
  (need Paul [dollar:20]).
```

As a second demonstration our second conceptual graph translates to the following verbatim :

```
insert
  [woman:Eva]
  (offer [sponsor:*] [asset:*a] Eva)
  (dream-about Eva [event:*e])
  (performer [show:*s] [clown:*c])
  (in ?s ?e)
  (in ?c ?e)
  (finance [funds:*f] ?e)
  (play [man:*m] ?c)
  (in ?f ?a)
  (in ?m ?a).
```

Of course both demonstrations build upon concept / relation hierarchies.
In the first case the hierarchies verbatim could be :

```
  untyped hierarchy entity relation.

  derive entity man money.
  derive man beggar.
  derive money dollar euro yen.

  derive1 relation1 wealth.
  derive2 relation2 need.
  derive3 relation3 give.
  derive1 wealth poor modest rich.
```

In the second case the hierarchies verbatim could be :

```
  untyped hierarchy entity relation.

  derive entity man woman sponsor asset event clown funds show.
  derive2 relation2 in dream-about performer finance play.
  derive3 relation3 offer.
```

A much more substantial example is this command line:

```
./ERic <anatomy-of-a-toy.cgif
```

10. Other ERic commands

select *cgif-graph***.** Find all `is-a` homomorphisms from the *cgif-graph* to the knowledge database.
quit. Quit the program.
save "*filename***".** Save the database to the binary *filename*.
load "*filename***".** Erase all and load the database from the binary *filename*.
derive4 relation4 *relations***.** Add new quaternary *relations* to the dictionary.

Along with signed integers ERic accepts signed floats and string constants.

11. ERic community & forum

The official ERic community & forum is kindly hosted at www.developpez.net

The official ERic SVN repository.

12. Going further

ERic v0.3 is merely a Knowledge Representation toy written in less than 1300 lines to demonstrate the OCaml programming language. If you need a full-fledged KR tool i recommend you to install CoGui 3.1 by the GraphiK team at LIRMM, Montpellier.