# Conceptual Structures

The causal behaviour of **concepts** is explained by their internal **structure**.

**Motives**

*Interface with Language*

| | |
|---|---|
| She broke the cup. | The cup broke. |
| She cut the bread. | The bread cut. (!) |

The information that guides us to form adequate **combinations** using a lexical concept `c` must be part of the structure of `c`.

*Interface with Reasoning*

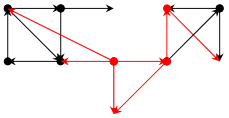
| | |
|---|---|
| She obliged him to sing. | He sang. |
| She allowed him to sing. | He sang. (?) |

The information that guides us to draw adequate **inferences** based on a lexical concept `c` must be part of the structure of `c`.

**Methods**

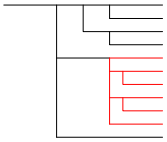
*Relational Method*

The structure of a lexical concept `c` is a portion of the **graph** of relations that link `c` to other structures.



*Definitional Method*

The structure of a lexical concept `c` is a definition within the **lattice** of previously defined structures.





# Definitions

$$\texttt{kill(x,y)} \equiv_{\text{déf}} \texttt{cause(x,e) \& die(e,y).}$$

$$\texttt{kill(x,y)} \equiv_{\text{déf}} \texttt{cause(x,e) \& path(e,y,d,f) \& life(d) \& death(f).}$$

$$\texttt{die(x)} \equiv_{\text{déf}} \texttt{path(x,d,f) \& life(d) \& death(f).}$$

Various ways of representing the meaning of “kill” symbolically.

## Functional structures

kill(x,y)=$_{déf}$ cause(x,path(y,life,death)).

kill(x,y)=$_{déf}$ cause(x,die(y)).
die(x)=$_{déf}$ path(x,life,death).

cause =$_{déf}$ λP.λx.cause(P)(x)::np\s/s.
die =$_{déf}$ λx.die(x)::np\s.
kill =$_{déf}$ λy.λx.cause(die(y))(x)::np\s/np.



## Functional structures

Synchronizing syntax
(here, categorial grammar)
and semantics (here lambda calcul).

cause =$_{déf}$ λP.λx.cause(P)(x)::np\s/s.
die =$_{déf}$ λx.die(x)::np\s.
kill =$_{déf}$ λy.λx.cause(die(y))(x)::np\s/np.

A::np  kill::np\s/np  B::np
A::np  (kill::np\s/np)(B::np)
A::np  (λy.λx.cause(die(y))(x)::np\s/np)(B::np)
A::np  λx.cause(die(B))(x)::np\s
(λx.cause(die(B))(x)::np\s)(A::np)
cause(die(B))(A)::s

## Type-based definitions

Typing to constrain meaning.

$\texttt{kill(x,y)}\rightarrow_{\text{déf}}$ $\texttt{[}_{\text{event}}\ \texttt{cause([}_{\text{entity}}\ \texttt{x],}$

$\texttt{[}_{\text{event}}\ \texttt{path([}_{\text{entity}}\ \texttt{y],[}_{\text{location}}\ \texttt{life],[}_{\text{location}}\ \texttt{death])])].}$

$\texttt{kill(x,y)}\rightarrow_{\text{déf}}$ $\texttt{[}_{\text{event}}\ \texttt{cause([}_{\text{entity}}\ \texttt{x],[}_{\text{event}}\ \texttt{die([}_{\text{entity}}\ \texttt{y])])].}$

$\texttt{die(x)}\rightarrow_{\text{déf}}$ $\texttt{[}_{\text{event}}\ \texttt{path([}_{\text{entity}}\ \texttt{x],[}_{\text{location}}\ \texttt{life],[}_{\text{location}}\ \texttt{death])].}$

$\texttt{[}_{\text{event}}\ \texttt{cause([}_{\text{entity}}\ \texttt{x],[}_{\text{event}}\ \texttt{i])].}$

$\texttt{[}_{\text{event}}\ \texttt{path([}_{\text{entity}}\ \texttt{x],[}_{\text{location}}\ \texttt{u],[}_{\text{location}}\ \texttt{v])].}$



## Lexical conceptual structure (LCS)

*place(Sonia,book) & on(book,table)*
*place(Sonia,book,P) & situation(P,on,table)*
*place(Sonia,book,on(table))*

$$\left[ {}_{\text{Evé nement}}\,CAUSE\left( \left[ {}_{\text{Objet}}\ Sonia \right], \left[ {}_{\text{Evé nement}}\,GO\left( \left[ {}_{\text{Objet}}\ livre \right], \left[ {}_{\text{Chemin}}\ TO\left( \left[ {}_{\text{Place}}\ ON\left( \left[ {}_{\text{Objet}}\ table \right] \right) \right] \right) \right] \right) \right] \right)$$

## Lexical conceptual structure (LCS)

$$[\texttt{Catégorie}] \rightarrow \begin{bmatrix} \texttt{Chose / Evénement / Action...} \\ \textit{Objet / type} \\ F\left(\langle \texttt{Catégorie}_1, \langle \texttt{Catégorie}_2, \langle \texttt{Catégorie}_3 \rangle \rangle \right) \end{bmatrix}$$

$$[\texttt{Place}] \rightarrow \left[ {}_{\texttt{Place}}\, \textit{Fonction\_localisatrice}([\texttt{Chose}]) \right]$$

$$[\texttt{Evénement}] \rightarrow \left\{ \begin{array}{l} \left[ {}_{\texttt{Evénement}}\, \textit{GO}([\texttt{Chose}], [\texttt{Chemin}]) \right] \\ \left[ {}_{\texttt{Evénement}}\, \textit{STAY}([\texttt{Chose}], [\texttt{Place}]) \right] \end{array} \right\}$$

$$[\texttt{Evénement}] \rightarrow \left[ {}_{\texttt{Evénement}}\, \textit{CAUSE}\left( \left[ \begin{Bmatrix} \texttt{Chose} \\ \texttt{Evénement} \end{Bmatrix} \right], [\texttt{Evénement}] \right) \right]$$

$$[\texttt{Chemin}] \rightarrow \left[ {}_{\texttt{Chemin}} \begin{Bmatrix} \textit{FROM} \\ \texttt{TO} \\ \textit{TOWARD} \\ \ldots \end{Bmatrix} \left( \left[ \begin{Bmatrix} \texttt{Chose} \\ \texttt{Place} \end{Bmatrix} \right] \right) \right]$$



## Lexical conceptual structure (LCS)

$$\begin{bmatrix} \texttt{beurrer} \\ \texttt{V} \\ \texttt{——NP}_j \end{bmatrix}$$

$$\left[ {}_{\texttt{Evénement}}\, \textit{CAUSE}\left( [\texttt{Chose}\ ]_i, \left[ {}_{\texttt{Evénement}}\, \textit{GO}\left( [\texttt{Chose}\ \textit{BUTTER}], \left[ {}_{\texttt{Chemin}}\, \textit{TO}\left( [\texttt{Place}\ \textit{ON}\left( [\texttt{Chose}\ ]_j \right) ] \right) \right] \right) \right] \right)$$

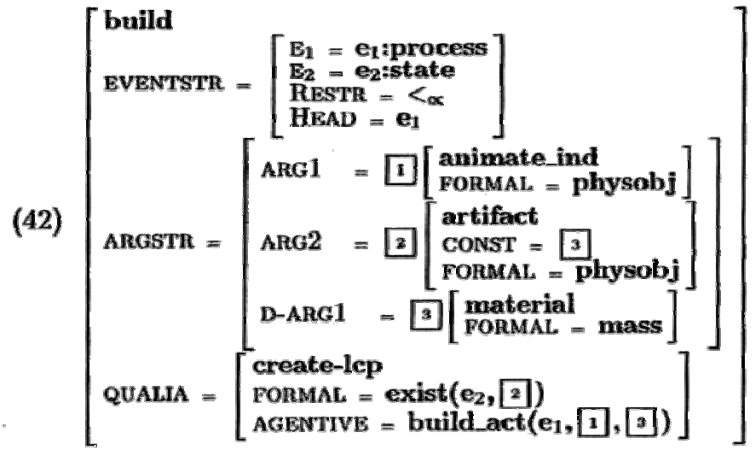
$$\left[ \textit{Evé nement}\, GO\left( [X], \left[ \textit{Che}\min\, TO([Y]) \right] \right) \right] \longrightarrow \left[ \textit{Etat}\, BE\left( [X], \left[ \textit{Place}\, AT([Y]) \right] \right) \right]$$

$$\begin{bmatrix} X^0 \\ \texttt{——}\langle YP \langle ZP \rangle \rangle \end{bmatrix} \longleftrightarrow \begin{bmatrix} \texttt{Catégorie} \\ F\left( \langle E_1 \rangle, \langle E_2, \langle E_3 \rangle \rangle \right) \end{bmatrix}$$

syntagme LCS

Techniques like HSPG process syntactic and semantic features similarly.

(42)






## Feature structures in ALE

Techniques like the "attribute logic engine" process syntactic and semantic features similarly.

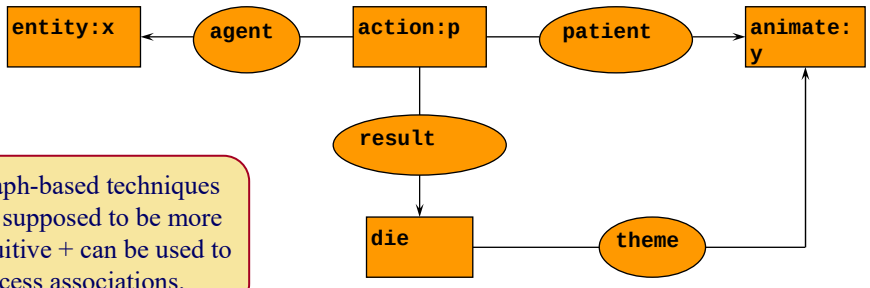
```
predicate subtypes   [unary,binary]
    features   [séquence : séquence].
unary  subtypes   []
    features   [argument : entity].
binary subtypes   []
    features   [argument1 : entity,
         argument2 : entity].

séquence subtypes []
    features   [step1 : situation,
         step2 : situation,
         step3 : situation].

situation subtypes   [state,event,processus,nul]
    features   [theme : entity].
state  subtypes   [].
event  subtypes   [].
processus subtypes   []
    features   [cause : entity].

entity subtypes   [abstract,concrete].
concrete   subtypes   [object,location].
object subtypes   [animate,inanimate].
```

```
kill(e1,e2)⇒ déf
    séquence:  step1:  theme:  e2,
               cause:  e1,
          step2:  event,  theme:  e2,
          step3:  nul,
       argument1:  e1,
       argument2:  e2,  animate.
```

*« A tue B »*

```
séquence:  step1:  theme:  B,
           cause:  A,
      step2:  event,  theme:  A,
      step3:  nul,
argument1:  A,
argument2:  B.
```

## Relational structures: semantic networks and conceptual graphs

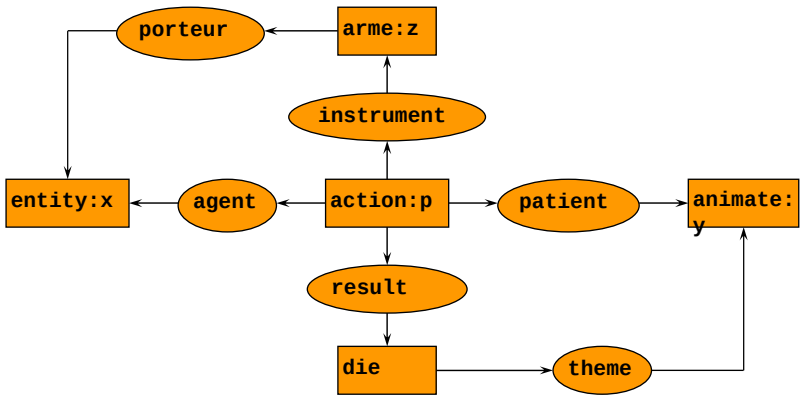



Graph-based techniques are supposed to be more intuitive + can be used to process associations.

```
[kill:p]: [action:p] & [entity:x] & [animate:y] & [die:q]
          agent(x,p)& patient(y,p) & result(p,q) & theme(y,q).
```



## Relational structures: semantic networks and conceptual graphs



## Relational structures : theories

$f = ma$
$P = mg$
$h = -gt^2/2$
$E_c = mv^2/2$
$W = f.x$
$E_p = mgh$
$p = mv$

$w = h\nu$
$H\psi = E\psi$
$E = mc^2$
$\Delta x.\Delta v > h/4\pi$

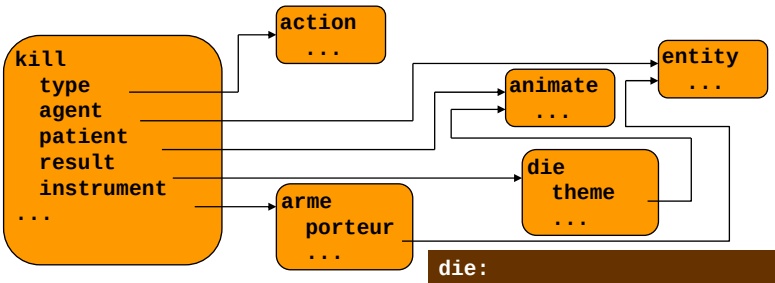
$\nabla.B = 0$
$\nabla.E = \rho/\varepsilon_0$
$\nabla \times B = \mu_0 j + \varepsilon_0 \partial E/\partial t$
$\nabla \times E = -\partial B/\partial t$

laiton on cuivre → brasure
brasure → T°> 800°
chauffage → non étain
brasure → apport_métal

"theories" are meant to capture local knowledge.



## Relational structures : frames




```
die:
        type → event;
        theme → entity (1);
        antérieur → [life:
                type → state;
                theme → entity (1)];
        postérieur → [death:
                type → state;
                theme → entity (1)];
        circonstance → vieillesse;
        etc.
```

Frames combine *features* and *procedures* (gave rise to object–oriented languages)