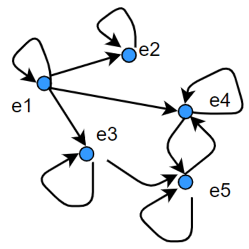
Ordre

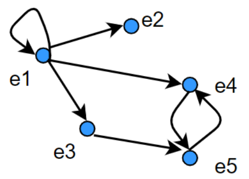
R ⊆ E × F (R relation binaire, E et F deux ensembles)

$$E_v = \{Paris, Berlin, Rome, Montpellier\}$$
$$F_p = \{Allemagne, France, Italie, Espagne\}$$
$$R_{vp} = \{(Paris, France), (Berlin, Allemagne), (Rome, Italie)\}$$

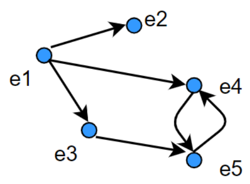
Une relation réflexive



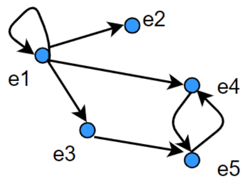
Une relation non réflexive



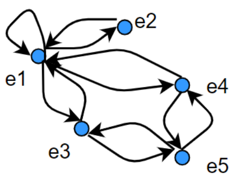
Une relation irréflexive



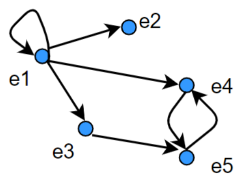
Une relation non irréflexive



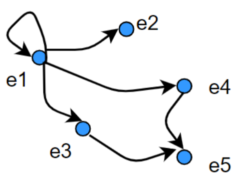
Une relation symétrique



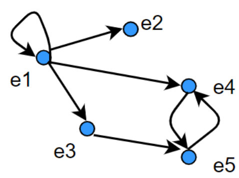
Une relation non symétrique



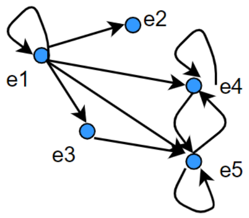
Une relation antisymétrique



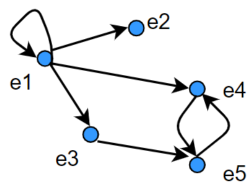
Une relation non antisymétrique



Une relation transitive



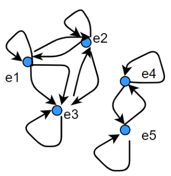
Une relation non transitive



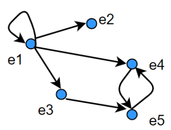
Relation d'équivalence :

- réflexive
- symétrique
- transitive

Une relation d'équivalence



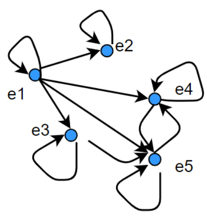
Une relation qui n'est pas une relation d'équivalence



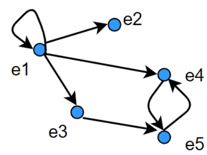
Pré-ordre :

- réflexive
- transitive

Un pré-ordre



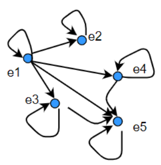
Une relation qui n'est pas un préordre



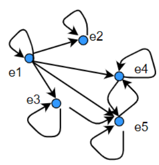
Ordre :

- réflexive
- antisymétrique
- transitive

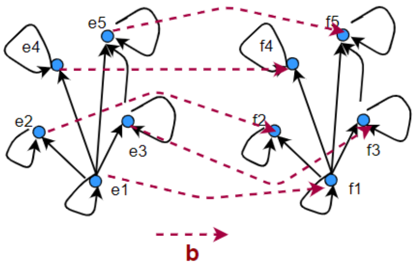
Un ordre

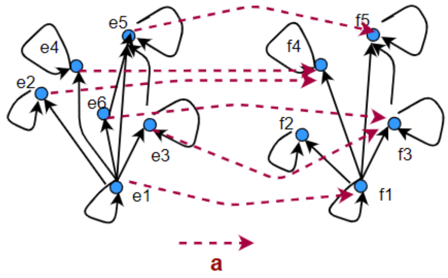


Une relation qui n'est pas un ordre

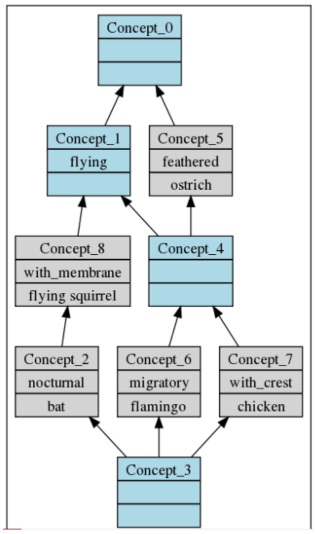


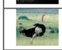
- Si e1 a plus d'arc entrants que e2 alors e2 ≤ e1 (dit majore)
- Si e1 et e2 ont un arc en commun alors ils sont comparables, sinon non
- e1 couvre e2 si e1 et le seul majorant de e2 (possible si e1 n'a pas d'arcs avec les autres majorants de e2)

<table>
<tr>
<td>Un isomorphisme d'ordre


Bijection entre les deux ens</td>
<td>

Morphisme non isomorphe : respecte l'ordre ≤</td>
</tr>
<tr>
<td>Produit cartésien de deux ensembles ou diagrammes E et F :
1. dessiner le second graphe x fois (avec x le nombre d'éléments dans le premier) en donnant la forme du premier (si E forme un carré de 4 elts alors faire 4 schéma F en forme de carré)
2. lier chaque elt des sous schéma suivant le schéma de E
3. lier chaque elt des sous schéma au même elt des schémas supérieurs</td>
<td>Construire le treilli :
1. mettre l'attribut le plus commun tout en haut (si tlm ne l'a pas rajouté un autre au dessus et le second plus commun à côté de l'autre si en additionnant les deux on a pas tlm ajouter le 3e)
2. ajouter en dessous les elts suivant moins communs, créer des elt tempo pour faire des combinaisons si besoin</td>
</tr>
<tr>
<td>Treillis :
Si elt qui majore tlm ➡ sup-demi-treilli
Si elt qui minore tlm ➡ inf-demi-treilli
si les deux alors treilli</td>
<td></td>
</tr>
</table>



	flying	nocturnal	feathered	migratory	with_crest	with_membrane
 flying squirrel	×					×
 bat	×	×				×
 ostrich			×			
 flamingo	×		×	×		
 chicken	×		×		×	

1. On ajoute le concept 0 car aucun atr n'est possédé par tlm
2. on ajoute l'atr le plus commun "flying" comme il ne suffit à avoir tlm on met le 2e "feathered" et ça suffit
3. l'un des elt ne possède que "feathered" on peut donc l'ajouter au treilli
4. en dessous on ajoute l'atr le plus commun donc "with_membrane" si tous les elt qui ne sont pas passés et ont flying et with membrane alors on relie l'atr à flying
5. on rajoute flying squirrel car il a tous les éléments
6. certains elt on feathered et flying, on crée donc un concept qui relie les 2
7. …

Opérateur de fermeture :

Exemple : fermeture en coordonnées entières

Soit l'ensemble des points dans le plan en coordonnées réelles $E = \mathbb{R} \times \mathbb{R}$, muni de l'ordre $\leq_E$ avec $(x_1, y_1) \leq_E (x_2, y_2)$ ssi $x_1 \leq x_2$ et $y_1 \leq y_2$
et $h : E \longrightarrow E$, avec $h(x, y) = (\lceil x \rceil, \lceil y \rceil)$
où $\lceil x \rceil$ est la partie entière supérieure de x
Exemple : $h(3.2, 6.8) = (4, 7)$

h est un opérateur de fermeture

- h est croissante : $(x_1, y_1) \leq_E (x_2, y_2) \Rightarrow (\lceil x_1 \rceil, \lceil y_1 \rceil) \leq_E (\lceil x_2 \rceil, \lceil y_2 \rceil)$
- h est extensive : $(x_1, y_1) \leq_E (\lceil x_1 \rceil, \lceil y_1 \rceil)$
- h est idempotente : $h(h((x_1, y_1))) = h((\lceil x_1 \rceil, \lceil y_1 \rceil)) = (\lceil x_1 \rceil, \lceil y_1 \rceil) = h((x_1, y_1))$

Elément fermé pour h

(x_1, y_1) est un élément fermé si x_1 et y_1 sont des entiers

Induction

Mult :

. Soit $mult : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

(m_1) $\forall n \in \mathbb{N} . mult(0, n) = 0$.
(m_2) $\forall p, n \in \mathbb{N} . mult((S\, p), n) = plus(mult(p, n), n)$.

Démontrer que : $\forall n \in \mathbb{N} . mult(2, n) = plus(n, n)$.

. On sait que $2 = (S\,(S\,0))$. Ainsi :

$$\begin{aligned} mult((S\,(S\,0)), n) &= plus(mult((S\,0), n), n) && \text{par } m_2 \\ &= plus(plus(mult(0, n), n), n) && \text{par } m_2 \\ &= plus(plus(0, n), n) && \text{par } m_1 \\ &= plus(n, n) && \text{par } plus(0, n) = n \end{aligned}$$

Démontrer que : $\forall n \in \mathbb{N} . mult(n, 2) = plus(n, n)$.

Lemme 1. $\forall n \in \mathbb{N} . plus(n, 0) = n$

Démonstration. On prouve le lemme 1 par induction structurelle :

Base Pour $n = 0$, on a $plus(0, 0) = 0$ par p_1. Par réflexivité, la propriété est vérifiée pour le cas de base.

Induction On suppose que $plus(n, 0) = n$. Montrons que $plus((S\,n), 0) = (S\,n)$.

$$\begin{aligned} plus((S\,n), 0) &= (S\, plus(n, 0)) && \text{par } p_2 \\ &= (S\,n) && \text{par hypothèse d'induction} \end{aligned}$$

Par réflexivité, la propriété est vérifiée $\forall n \in \mathbb{N}$.

Lemme 2. $\forall n, m \in \mathbb{N} . plus(m, (S\,n)) = (S\, plus(m, n))$

Démonstration. On prouve le lemme 2 par induction structurelle :

Base Pour $m = 0$, on a $plus(0, (S\,n)) = (S\,n)$ par p_1, et $(S\, plus(0, n)) = (S\,n)$. Par réflexivité,
$(S\,n) = (S\,n)$ et donc la propriété est vérifiée pour le cas de base.

Induction On suppose que la propriété est vraie pour un m quelconque, c'est à dire que $plus(m, (S\ n)) = (S\ plus(m, n))$. Montrons que $plus((S\ m), (S\ n)) = (S\ (S\ plus(m, n)))$.

$$plus((S\ m), (S\ n)) = (S\ plus(m, (S\ n))) \quad \text{par } p_2$$
$$= (S\ (S\ plus(m, n)) \quad \text{par hypothèse d'induction}$$

Par réflexivité, on a bien $plus((S\ m), (S\ n)) = (S\ (S\ plus(m, n)))$, donc la propriété est vérifiée
$\forall n, m \in \mathbb{N}$.

- On peut maintenant montrer $\forall n \in \mathbb{N}\ .mult(n, 2) = plus(n, n)$ par induction :

  Base Pour $n = 0$, on a $mult(0, 2) = 0$ par m_1, et $plus(0, 0) = 0$ par p_1. Par réflexivité, $mult(0, 2) = plus(0, 0)$ et ainsi la propriété est vérifiée pour le cas de base.

  Induction On suppose que $\forall\ n \in \mathbb{N}\ mult(n, 2) = plus(n, n)$.
  Montrons que $\forall\ n \in \mathbb{N}\ mult((S\ n), 2) = plus((S\ n), (S\ n))$. On sait que $2 = (S\ (S\ 0))$.

$$mult((S\ n), (S\ (S\ 0))) = plus(mult(n, 2), (S\ (S\ 0))) \quad \text{par } m_2$$
$$= plus(plus(n, n), (S\ (S\ 0))) \quad \text{par hypothèse d'induction}$$
$$= (S\ plus(plus(n, n), (S\ 0))) \quad \text{par le lemme 2}$$
$$= (S\ (S\ plus(plus(n, n), 0))) \quad \text{par le lemme 2}$$
$$= (S\ (S\ plus(n, n))) \quad \text{par le lemme 1}$$

De plus :

$$plus((S\ n), (S\ n)) = (S\ plus(n, (S\ n))) \quad \text{par } p_2$$
$$= (S\ (S\ plus(n, n))) \quad \text{par le lemme 2}$$

Par réflexivité, on conclut que $mult((S\ n), 2) = plus((S\ n), (S\ n))$ et donc la propriété est vérifiée $\forall n \in \mathbb{N}$.

Rev :

$rev : \mathrm{L} \to \mathrm{L}$:
$(l_1)\ rev([\ ]) = [\ ]$
$(l_2)\ \forall e \in \mathrm{A},\ \forall l \in \mathrm{L}.rev(e :: l) = app(rev(l), [e])$.

is_rev : L x L -> Prop
1. is_rev(nil, nil)

3. Va E A . Vl1, l2 E L . is_rev(l1, l2) -> is_rev(a::l1, app(l2, a::nil))

[3,2,1] [1,2,3]

R3 a = 3, is_rev([2,1], [1,2]) l1 = [2,1] l2 = [1,2], a::l1 = 3::[2,1]

is_rev([2,1], [1,2]) ?

Schéma d'induction fonctionnelle:

$\underbrace{\text{P(nil, nil)}}_{\text{cas de base}}$ -> $\underbrace{\text{(Vl1,l2 E L . Va E A . P(l1, l2) -> P(a::l1, app(l2, [a]))}}_{\text{induction P(n)->P(n+1)}}$ -> $\underbrace{\text{(Vl1,l2 E L . P(l1, l2))}}_{\text{conclusion}}$

Correction : si ma fonction donne un résultat, ce résultat est correct

Cas de base:

rev(nil) = nil est vrai (R1) de rev

is_rev(nil, nil) est vrai (R1) de is_rev

T -> T ce qui est vrai   donc P(nil, nil) = (rev(nil) = nil -> is_rev(nil, nil))

Hypothèse : P(l1, l2) = "rev(l1) = l2 -> is_rev(l1, l2) "

cons:

rev(l1) = l2

(R2) de rev: rev(a::l1) = app(rev(l1), [a])   l2

is_rev(a::l1, app(l2, [a])

l2 = rev(l1) (HR)

(R2) de is_rev : is_rev(l1, l2) (HI) -> is_rev(a::l1, app(l2, [a])

Donc is_rev(a::l1, app(l2, [a]), donc P(a::l1, app(l2, [a]))

Donc P(l1, l2)

rev

A rev(l1) = l2

-> rev(a::l1) = app(rev(l1), [a])

B rev(a::l1) = app(l2, [a])

is_rev

C is_rev(l1, l2)

R2 : is_rev(l1, l2) -> is_rev(a::l1, app(l2, [a]))

D

Relation inductive : Spécifie le comportement de la fonction
pour prouver, partir de 1) et avancer

is_fact : N x N → Prop   S0 = succ 0

1) is_fact(0,S0)

2) ∀a,b∈N is_fact(a,b) → is_fact(Sa,mult(Sa,b))

Fonction : ex :

fact : N → N

1) fact(0) = S0

2) ∀a∈N fact(Sa) = mult(Sa, fact(a))

Schéma d'induction fonctionnelle :

Schéma à savoir par coeur: (0) → (n → Sn) → n

(∀p∈N, fact(0) = p → is_fact(0,p)) → [∀n,p∈N (fact(n)=p → is_fact(n,p)) → (fact(Sn) = mult(Sn, p) → is_fact(Sn, mult(Sn,p)))] → (∀n,p∈N fact(n)=p → is_fact(n,p))

Correction :

Base

fact(0) = 1 -- (fact-1)

is_fact(0,1) -- (is_fact-1)

Donc fact(0)=1 -> is_fact(0,1)

Construction

On suppose que ∀n,p∈N fact(n)=p -> is_fact(n,p)

fact(Sn) = mult(Sn, fact(n)) -- (fact-2)

is_fact(n,p) -> is_fact(Sn, mult(Sn, p)) -- (is_fact-2)

On remplace p par fact(n):

is_fact(n,p) -> is_fact(Sn, mult(Sn, fact(n)))

Donc ∀n,p∈N (fact(n)=p -> is_fact(n,p)) -> (fact(Sn) = mult(Sn,p) -> is_fact(Sn, mult(Sn, fact(n)))) par modus ponens.

Donc d'après le schéma d'induction fonctionnelle, ∀n,p∈N fact(n)=p -> is_fact(n,p).