

# La syntaxe dynamique par Ruth Kempson

Nicolas GUILLIOT

11 juillet 2005

Basé sur Cann & Kempson (2004), *The dynamics of Language*.

But : Créer un modèle dynamique qui puisse rendre compte de :

-l'interprétation en contexte (mot-à-mot)

-généralisations syntaxiques (notamment sur les dépendances à longue distance).

## 1 Théorie des types & application fonctionnelle

Voir Heim & Kratzer (1998) pour plus de détails.

L'analyse des propositions (et le calcul du sens) obéit à une théorie des types : idée que tous les éléments de la langue ne vont pas dénoter la même chose.

Distinction entre termes singuliers, prédicats, et propositions.

### 1.1 Les entités individuelles

Un terme singulier réfère à une entité individuelle du monde  $\rightarrow$  type  $e$  (pour entity)

(1)  $\llbracket \text{Jean} \rrbracket =$  l'individu 'Jean' dans le monde,  $\in D_e$

### 1.2 Les propositions

Une proposition va dénoter une valeur de vérité (1 si la phrase est vraie, 0 si la phrase est fausse)  $\rightarrow$  type  $t$ .

(2)  $\llbracket \text{Jean regarde la télé} \rrbracket = 1$  ssi Jean regarde la télé, sinon 0,  $\in D_t$

### 1.3 Les prédicats

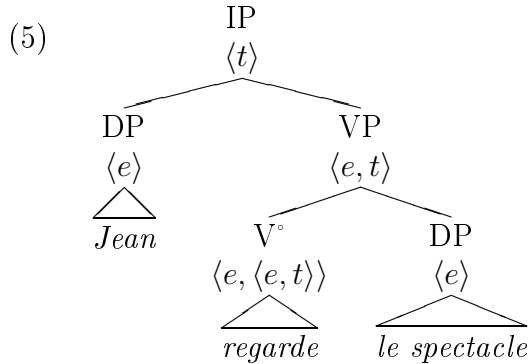
Un prédicat dénote une fonction qui s'applique à des individus  $\rightarrow$  type  $\langle e, t \rangle$  si 1 individu ou argument, type  $\langle e, \langle e, t \rangle \rangle$  si 2 individus.

(3)  $\llbracket \text{regarder} \rrbracket = \lambda y \in D_e [\lambda x \in D_e . x \text{ regarde } y], \in D_{\langle e, \langle e, t \rangle \rangle}$

(4)  $\llbracket \text{donner} \rrbracket = \lambda z \in D_e [\lambda y \in D_e [\lambda x \in D_e . x \text{ donne } z \text{ à } y]], \in D_{\langle e, t \rangle}$

## 1.4 combinaison des types : application fonctionnelle

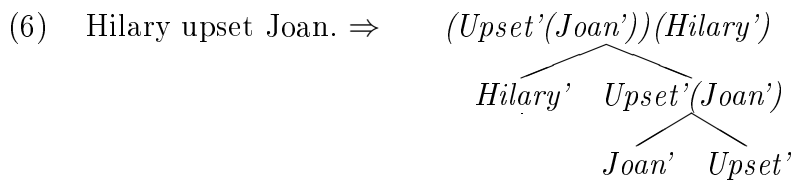
La théorie des types détermine les combinaisons possibles entre prédicats, entités individuelles et propositions :



Un prédicat de type  $\langle e, \langle e, t \rangle \rangle$  comme *regarder* va se combiner avec une première entité individuelle  $\langle e \rangle$  pour former le VP (1ère application fonctionnelle), puis avec une seconde entité individuelle (2ème application fonctionnelle) pour donner une valeur de vérité.

## 2 Le Modèle de la syntaxe dynamique

- Analyse incrémentale du langage, i.e. mot-à-mot (parsing).
- Représentation sous la forme d'arbres pour représenter les interprétations assignées aux mots en contexte.
- Les noeuds sont des concepts dérivés de la composition sémantique. Ils représentent la structure sémantique véhiculée par ces concepts.
- Les arbres en syntaxe dynamique ne traduisent pas l'ordre des mots, mais seulement la compositionnalité du sens  $\Rightarrow$  L'argument, ce qui modifie (ex : l'entité *Joan*), est toujours placé à gauche, le foncteur, ce qui est modifié (ex : le prédicat *upset*), à droite.



Techniquement, le système fonctionne de la manière suivante : l'arbre se construit et, au fur et à mesure, le lexique (sous la forme de petits programmes) y est inséré. Maintenant, nous allons voir tout ce qui va décorer les noeuds.

### 2.1 Informations sur les noeuds

#### 2.1.1 types et formules

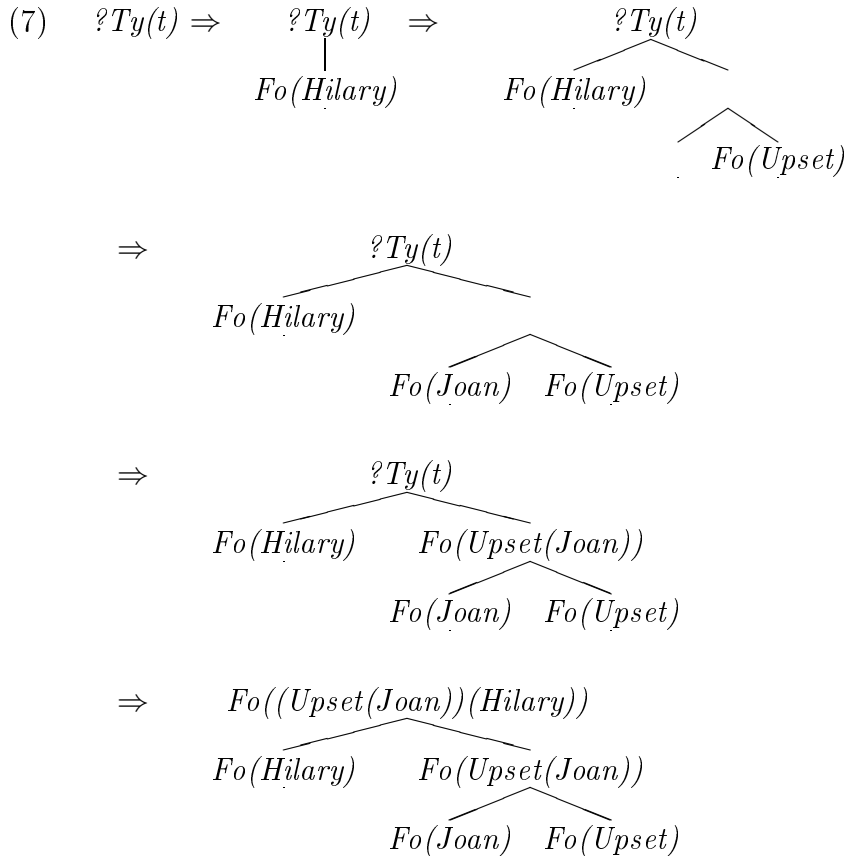
Information sémantique : un type et une formule.

Type	Description	Formule/concept
$Ty(e)$	Entité	$Fo(Marie')$
$Ty(t)$	Proposition	$Fo(Sing'(John')), Fo(Upset'(Hilary'))(Joan')$
$Ty(e \rightarrow t)$	Prédicat 1 place	$Fo(Run')$
$Ty(e \rightarrow (e \rightarrow t))$	Prédicat 2 places	$Fo(Upset')$
$Ty(e \rightarrow (e \rightarrow (e \rightarrow t)))$	Prédicat 3 places	$Fo(Give')$
$Ty(t \rightarrow (e \rightarrow t))$	Prédicat	$Fo(Believe')$
$Ty(cn)$	Nom	$Fo(x, Etudiant(x))$

### 2.1.2 Point de départ : requête et développement de l'arbre

Les noeuds peuvent contenir des requêtes. La notion de requête est basée d'abord sur l'idée que, lorsqu'on s'exprime, on veut apporter une information et construire une proposition (une affirmation).

Notation utilisée :  $?Ty(t)$  (signifie que le système recherche un élément de type  $t$ .)



$\Rightarrow$  L'arbre se construit bien mot-à-mot jusqu'à satisfaction de la ou des requêtes.

### 2.1.3 Localisation et pointeur

Le système étant dérivationnel, il faut un pointeur qui désigne le noeud où les actions sont en cours.

Notation utilisée :  $\diamond$



## 2.2 Construction d'arbre et Lexique

Le système comporte 2 types d'opérations.

Une première partie gère la construction de l'arbre à travers des règles de composition, créant ainsi des requêtes et créant surtout la motivation pour l'insertion du lexique.

La deuxième partie est le lexique lui-même, représenté sous forme de petits programmes qui vont venir modifier l'arbre en ajoutant notamment la contribution sémantique.

### 2.2.1 Le lexique

Le lexique (les mots) se présente sous la forme de petits programmes qui vont permettre d'annoter les noeuds des arbres pré-construits par des règles.

Notation utilisée :	$mot'$	IF $?Ty(X)$ THEN ... ELSE ...	Motivation Actions Autre
---------------------	--------	-------------------------------------	--------------------------------

L'insertion du lexique sera motivée par une requête ('if'), ce qui déclenchera sur l'arbre des actions du type go (aller à un noeud), make (créer un noeud), put (annoter un noeud). Exemples pour *Hilary* et *Upset* :

$Hilary'$	IF $?Ty(e)$ THEN $put(Ty(e), Fo(Hilary'), [\downarrow] \perp)$ ELSE ABORT	Motivation Actions Autre
$Upset'$	IF $?Ty(e \rightarrow t)$ THEN $go(\langle \uparrow_1 \rangle ?Ty(t))$ , $put(Tns(PAST))$ , $go(\langle \downarrow_1 \rangle ?Ty(e \rightarrow t))$ , $make(\langle \downarrow_1 \rangle)$ , $put(Fo(Upset'), Ty(e \rightarrow (e \rightarrow t)), [\downarrow] \perp)$ ; $go(\langle \uparrow_1 \rangle)$ ; $make(\langle \downarrow_0 \rangle)$ ; $go(\langle \downarrow_0 \rangle)$ ; $put(?Ty(e))$ ELSE ABORT	Motivation Aller au noeud mère Information du temps Aller au noeud prédicat Créer un noeud foncteur Annotation Aller au noeud mère Créer un noeud argument Aller au noeud argument Annotation

### 2.2.2 Règles de construction de l'arbre

Les règles de développement de l'arbre sont appelées règles de transition. Elles sont toutes optionnelles, mais vont pouvoir activer l'insertion du lexique.

Notation utilisée :  $\frac{Premisse}{Conclusion}$

$$\text{-Introduction : } \frac{\{...\text{?Ty}(Y)\dots\Diamond\}}{\{\dots\text{?Ty}(Y), \text{?}\langle\downarrow_0\rangle\text{Ty}(X), \text{?}\langle\downarrow_1\rangle\text{Ty}(X \rightarrow Y), \dots\Diamond\}}$$

$$\text{-Prédiction : } \frac{\{\{Tn(n), \dots, \text{?}\langle\downarrow_0\rangle\phi, \text{?}\langle\downarrow_1\rangle\psi, \Diamond\}\}}{\{\{Tn(n), \dots, \text{?}\langle\downarrow_0\rangle\phi, \text{?}\langle\downarrow_1\rangle\psi\}, \{\langle\uparrow_0\rangle Tn(n), \text{?}\phi, \Diamond\}, \{\langle\uparrow_1\rangle Tn(n), \text{?}\psi\}\}}$$

Ces deux règles permettent par exemple de décliner la requête d'une proposition ( $Ty(t)$ ) en sous-requêtes :

$$\text{-Introduction sujet-prédicat : } \frac{\{\text{?Ty}(t), \Diamond\}}{\{\dots\text{?Ty}(t), \text{?}\langle\downarrow_0\rangle\text{Ty}(e), \text{?}\langle\downarrow_1\rangle\text{Ty}(e \rightarrow t), \Diamond\}}$$

$$\text{-Prédiction sujet-prédicat : } \frac{\{\{Tn(0), \text{?}\langle\downarrow_0\rangle\text{Ty}(e), \text{?}\langle\downarrow_1\rangle\text{Ty}(e \rightarrow t), \Diamond\}\}}{\{\{Tn(0), \text{?}\langle\downarrow_0\rangle\text{Ty}(e), \text{?}\langle\downarrow_1\rangle\text{Ty}(e \rightarrow t)\}, \{\langle\uparrow_0\rangle Tn(0), \text{?Ty}(e), \Diamond\}, \{\langle\uparrow_1\rangle Tn(0), \text{?Ty}(e \rightarrow t)\}\}}$$

On obtient alors l'arbre suivant :

$$(11) \quad \text{?Ty}(t), \text{?}\langle\downarrow_0\rangle\text{Ty}(e), \text{?}\langle\downarrow_1\rangle\text{Ty}(e \rightarrow t), \Diamond$$

$$\Rightarrow \quad \begin{array}{c} \text{?Ty}(t), \text{?}\langle\downarrow_0\rangle\text{Ty}(e), \text{?}\langle\downarrow_1\rangle\text{Ty}(e \rightarrow t) \\ \swarrow \quad \searrow \\ \text{?Ty}(e), \Diamond \quad \text{?Ty}(e \rightarrow t) \end{array}$$

Il existe d'autres règles (qui servent essentiellement à supprimer les annotations quand elles ne sont plus utiles) :

- Achèvement(Completion)  $\rightarrow$  pour remonter le pointeur et l'information ;
- Anticipation  $\rightarrow$  pour descendre le pointeur sur une requête saillante ;
- Dilution(Thinning)  $\rightarrow$  pour supprimer les requêtes satisfaites ;
- Elimination  $\rightarrow$  pour opérer sur le noeud mère l'application fonctionnelle des formules de ses 2 noeuds fille.

### 2.2.3 Une phrase, enfin !!!

Comment fonctionne le modèle sur une phrase comme *Hilary upset Joan* ?

-Introduction & Prédiction  $\Rightarrow$  création de l'arbre en (11).



- Elimination  $\Rightarrow$  
$$\begin{array}{c} Tns(PAST), ?Ty(t), ? \langle \downarrow_1 \rangle Ty(e \rightarrow t) \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Hilary') \end{array} \quad \begin{array}{c} ?Ty(e \rightarrow t), Ty(e \rightarrow t), \diamond, \\ \langle \downarrow_0 \rangle Ty(e), Fo(Upset'(Joan')) \end{array} \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Joan') \end{array} \quad \begin{array}{c} Ty(e \rightarrow (e \rightarrow t)), \\ Fo(Upset'), [\downarrow] \perp \end{array} \end{array}$$
- Dilution+ Achèvement  $\Rightarrow$  
$$\begin{array}{c} Tns(PAST), ?Ty(t), ? \langle \downarrow_1 \rangle Ty(e \rightarrow t), \langle \downarrow_1 \rangle Ty(e \rightarrow t), \diamond \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Hilary') \end{array} \quad \begin{array}{c} Ty(e \rightarrow t), \\ Fo(Upset'(Joan')) \end{array} \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Joan') \end{array} \quad \begin{array}{c} Ty(e \rightarrow (e \rightarrow t)), \\ Fo(Upset'), [\downarrow] \perp \end{array} \end{array}$$
- Dilution  $\Rightarrow$  
$$\begin{array}{c} Tns(PAST), ?Ty(t), \langle \downarrow_1 \rangle Ty(e \rightarrow t), \diamond \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Hilary') \end{array} \quad \begin{array}{c} Ty(e \rightarrow t), \\ Fo(Upset'(Joan')) \end{array} \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Joan') \end{array} \quad \begin{array}{c} Ty(e \rightarrow (e \rightarrow t)), \\ Fo(Upset'), [\downarrow] \perp \end{array} \end{array}$$
- Elimination  $\Rightarrow$  
$$\begin{array}{c} Tns(PAST), ?Ty(t), Ty(t), Fo(Upset'(Joan'))(Hilary'), \diamond \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Hilary') \end{array} \quad \begin{array}{c} Ty(e \rightarrow t), \\ Fo(Upset'(Joan')) \end{array} \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Joan') \end{array} \quad \begin{array}{c} Ty(e \rightarrow (e \rightarrow t)), \\ Fo(Upset'), [\downarrow] \perp \end{array} \end{array}$$
- Dilution  $\Rightarrow$  
$$\begin{array}{c} Tns(PAST), Ty(t), Fo(Upset'(Joan'))(Hilary'), \diamond \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Hilary') \end{array} \quad \begin{array}{c} Ty(e \rightarrow t), \\ Fo(Upset'(Joan')) \end{array} \\ \swarrow \quad \searrow \\ \begin{array}{c} Ty(e), [\downarrow] \perp, \\ Fo(Joan') \end{array} \quad \begin{array}{c} Ty(e \rightarrow (e \rightarrow t)), \\ Fo(Upset'), [\downarrow] \perp \end{array} \end{array}$$

#### 2.2.4 la dislocation : la règle d'adjonction

Pour pouvoir rendre compte de phrases avec topicalisation ou dislocation, il suffit d'ajouter une règle (non-réursive) qui permet de traiter un mot sans le fixer dans

la structure (notion de sous-spécification) :

$$\text{*Adjonction : } \frac{\{\{Tn(a), \dots ?Ty(t), \diamond\}\}}{\{\{Tn(a), \dots, ?Ty(t)\}, \{\langle \uparrow_* \rangle Tn(a), ?\exists x Tn(x), ?Ty(e), \diamond\}\}}$$

Cette règle va créer l'arbre suivant :

$$(12) \quad ?Ty(t), \diamond \Rightarrow \begin{array}{c} Tn(a), ?Ty(t) \\ | \\ \langle \uparrow_* \rangle Tn(a), ?Ty(e), ?\exists x.Tn(x), \diamond \end{array}$$

On peut maintenant traiter une phrase comme *Joan, Hilary upset* :

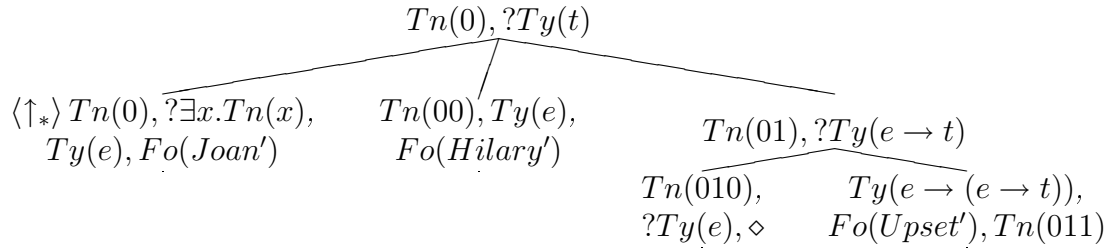
-Adjonction :  $\Rightarrow$  création de l'arbre en (12), avec  $a=0$

$$\text{-Insertion } Joan \Rightarrow \begin{array}{c} Tn(0), ?Ty(t) \\ | \\ \langle \uparrow_* \rangle Tn(0), ?Ty(e), ?\exists x.Tn(x), Ty(e), Fo(Joan'), [\downarrow] \perp, \diamond \end{array}$$

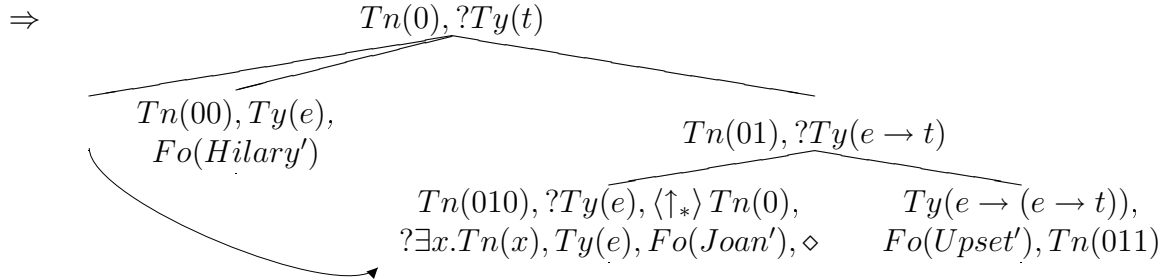
-Dilution+ Achèvement  $\Rightarrow$   $?Ty(e)$  est satisfaite et le pointeur remonte ;

-Insertion de *Hilary* et *Upset* : voir section 2.2.3

$\Rightarrow$

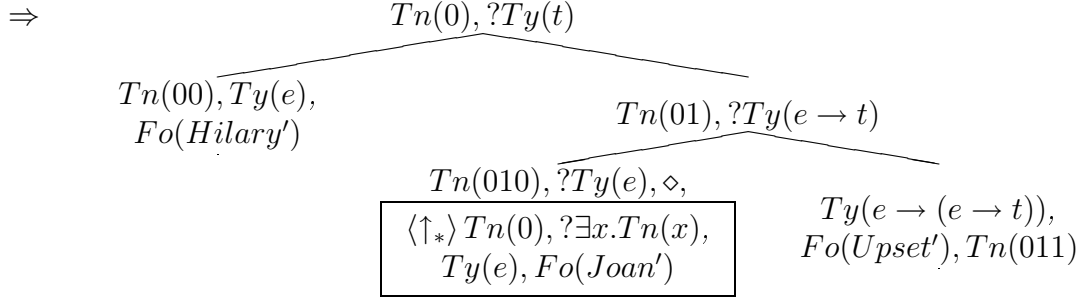


-Unification du noeud sous-spécifié avec celui de la requête  $?Ty(e)$  :

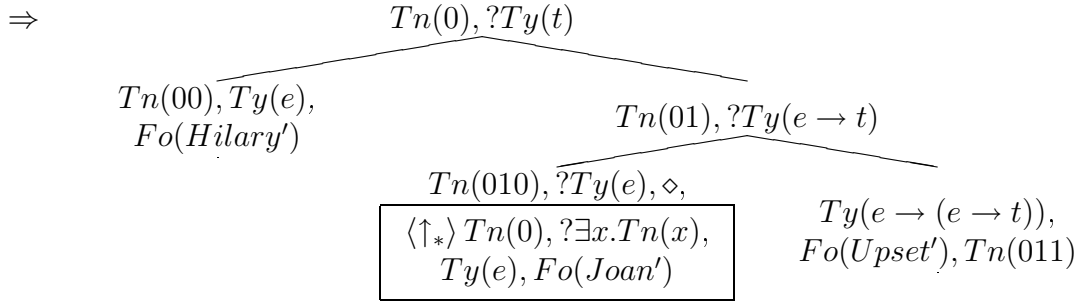


Techniquement, l'information portée par le noeud sous-spécifié se déplace sur tout noeud que rencontre le pointeur jusqu'à ce qu'il soit unifié avec l'un de ces noeuds (notion de *Merge*, mais différente de celle en GG) :

-Insertion de *Hilary* et *Upset* :



-Unification du noeud sous-spécifié avec celui de la requête  $?Ty(e)$  :



### 3 Anaphores, Liage et Localité

#### 3.1 Information lexicale des pronoms

En syntaxe dynamique, les pronoms sont définis par des formules de présupposition contenant des métavariabes ( $U$ ,  $V$ ) et par une requête pour que cette formule soit remplacée par Substitution.

(13)

$he$	IF	$?Ty(e)$	Motivation
	THEN	$put(Ty(e), Fo(U_{Male'}),$	Type et Formule de présupposition
		$?\exists x.Fo(x),$	Requête d'une formule
		$? \langle \uparrow \rangle (Ty(t) \wedge \exists y.Tns(y)),$	Condition de cas
		$[\downarrow] \perp$ ;	Noeud Terminal
	ELSE	ABORT	

La règle de Substitution va ensuite permettre d'identifier cette variable avec une formule du contexte (i.e.  $Fo(John)$ ).



La règle de Substitution va ainsi pouvoir inclure la condition B du liage :

$$(16) \quad Subst(\alpha) \left| \begin{array}{ll} \text{IF} & Fo(U), Ty(e) \\ \text{THEN IF} & \langle \uparrow_0 \rangle \langle \uparrow_1^* \rangle \langle \downarrow_0 \rangle Fo(\alpha) \\ & \text{THEN Abort} \\ & \text{ELSE put}(Fo(\alpha)) \\ \text{ELSE} & \text{Abort} \end{array} \right.$$

La condition A du liage sera, elle, définie directement dans l'information lexicale des pronoms réfléchis :

$$(17) \quad herself \left| \begin{array}{ll} \text{IF} & ?Ty(e) \\ \text{THEN IF} & \langle \uparrow_0 \rangle ?Ty(t) \\ & \text{THEN Abort} \\ & \text{ELSE IF} & \langle \uparrow_0 \rangle \langle \uparrow_1^* \rangle \langle \downarrow_0 \rangle Fo(\alpha) \\ & \text{THEN put}(Ty(e), Fo(\alpha); [\downarrow] \perp) \\ & \text{ELSE Abort} \\ & \text{ELSE Abort} \\ \text{ELSE} & \text{Abort} \end{array} \right.$$

## 4 Ordre des mots dans les langues : l'ordre VSO

Pour obtenir l'ordre VSO dans ce système, il suffit de modifier la description lexicale des verbes pour qu'ils puissent être traités dès la première requête  $?Ty(t)$ .

### Exemple en Irlandais :

- (18) *Chonaic mé an cú.*  
 voir-passé je le chien  
 'J'ai vu le chien'

La description du verbe aura la forme suivante :

$$Chonaic \left| \begin{array}{ll} \text{IF} & ?Ty(t) \\ \text{THEN} & \text{put}(Tns(PAST)) \quad \text{Temps} \\ & \text{make}(\langle \downarrow_1 \rangle); \text{go}(\langle \downarrow_1 \rangle); \text{put}(?Ty(e \rightarrow t)); \quad \text{Noeud prédicat} \\ & \text{make}(\langle \downarrow_1 \rangle); \text{go}(\langle \downarrow_1 \rangle); \\ & \text{put}(Fo(Chon'), Ty(e \rightarrow (e \rightarrow t)), [\downarrow] \perp) \quad \text{Foncteur principal} \\ & \text{go}(\langle \uparrow_1 \rangle); \text{make}(\langle \downarrow_0 \rangle); \text{go}(\langle \downarrow_0 \rangle); \text{put}(?Ty(e)) \quad \text{Argument interne} \\ & \text{go}(\langle \uparrow_0 \rangle \langle \uparrow_1 \rangle); \text{make}(\langle \downarrow_0 \rangle); \\ & \text{go}(\langle \downarrow_0 \rangle); \text{put}(?Ty(e)) \quad \text{Sujet} \\ \text{ELSE} & \text{Abort} \end{array} \right.$$

La syntaxe dynamique peut aussi rendre compte des langues pro-drop, en ajoutant à l'action du verbe une spécification du sujet.

**Exemple en Grec Moderne :**

- (19) *Ksero*                      *ti*            *Maria*.  
 connaître-passé-1sg    Maria  
 ‘Je connais Maria.’

La description du verbe aura la forme suivante :

<i>Ksero</i>	IF	$?Ty(t)$	
	THEN	$put(Tns(PRES))$ $make(\langle \downarrow_1 \rangle); go(\langle \downarrow_1 \rangle); put(?Ty(e \rightarrow t));$ $make(\langle \downarrow_1 \rangle); go(\langle \downarrow_1 \rangle);$ $put(Fo(Kser'), Ty(e \rightarrow (e \rightarrow t)), [\downarrow] \perp)$ $go(\langle \uparrow_1 \rangle); make(\langle \downarrow_0 \rangle); go(\langle \downarrow_0 \rangle); put(?Ty(e))$ $go(\langle \uparrow_0 \rangle \langle \uparrow_1 \rangle); make(\langle \downarrow_0 \rangle); go(\langle \downarrow_0 \rangle)$ $put(?Ty(e), Fo(U_{Speaker'}), \exists x.Fo(x))$	Temps Noeud prédicat Foncteur principal Argument interne Sujet
	ELSE	Abort	

Tout comme pour un pronom, la formule  $Fo(U_{Speaker'})$  va ensuite pouvoir être substituée par une autre formule (ex :  $Fo(Nico')$  si c'est moi qui parle)

## 5 Les relatives restrictives et appositives

### 5.1 Les relative appositives

Sémantiquement, une relative appositive introduit une seconde assertion (en plus de celle apportée par la proposition) qui se rapporte à une entité. On a donc deux assertions se rapportant à un même individu :

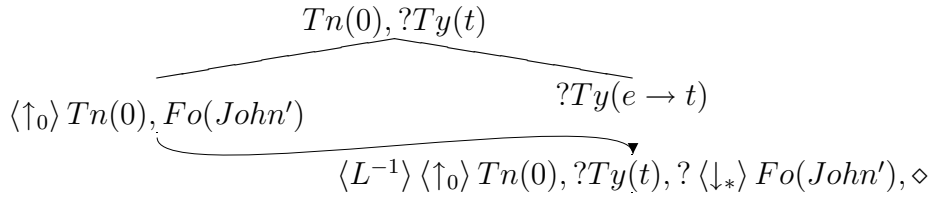
- (20) John, who I like, smokes.

La syntaxe dynamique va autoriser le développement partiel d'un arbre (proposition principale) suivi du développement d'un arbre subordonné (la relative appositive) avant de revenir à l'arbre principal. Plus précisément, la Syntaxe Dynamique va permettre de relier les 2 assertions à l'aide d'un opérateur de modalité, Lien (LINK), et d'une règle de transition, Adjonction de Lien :

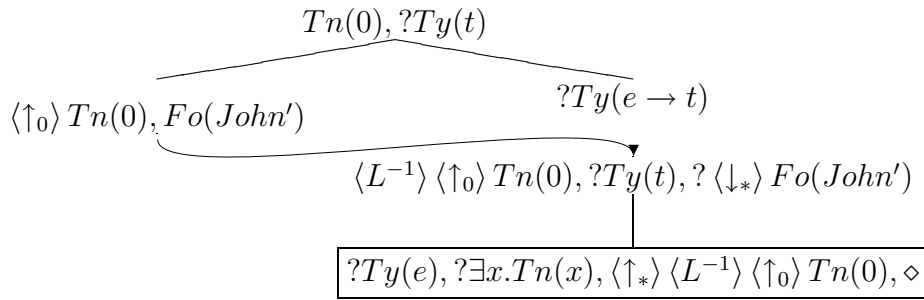
-Modalité de lien :  $\langle L \rangle$

-Adjonction de lien : 
$$\frac{\{... \{Tn(a), Fo(\alpha), ?Ty(e), \diamond\}\}}{\{... \{Tn(a), Fo(\alpha), ?Ty(e)\}, \{\langle L^{-1} \rangle Tn(a), ?Ty(t), ? \langle \downarrow_* \rangle Fo(\alpha), \diamond\}\}}$$

On commence par introduire *John*, puis à partir de ce noeud, la règle Adjonction de lien peut s'appliquer et va créer un arbre lié (qui se développe en parallèle) :



Cette structure permet d'introduire un noeud non-fixe comme pour la dislocation (adjonction) :

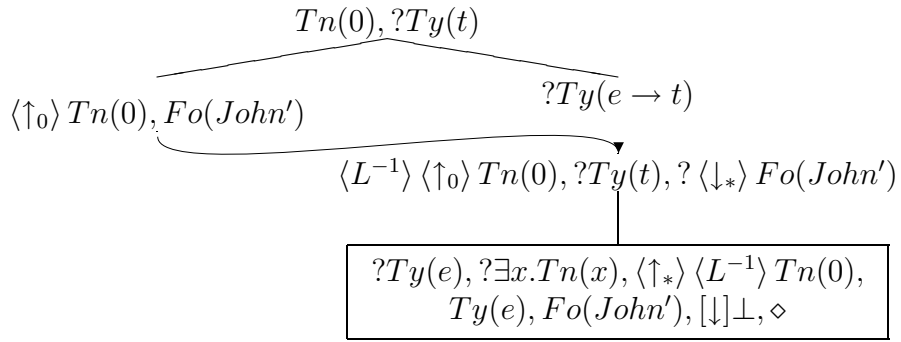


Il suffit maintenant d'avoir la bonne description lexicale pour les pronoms relatifs pour continuer l'analyse mot-à-mot :

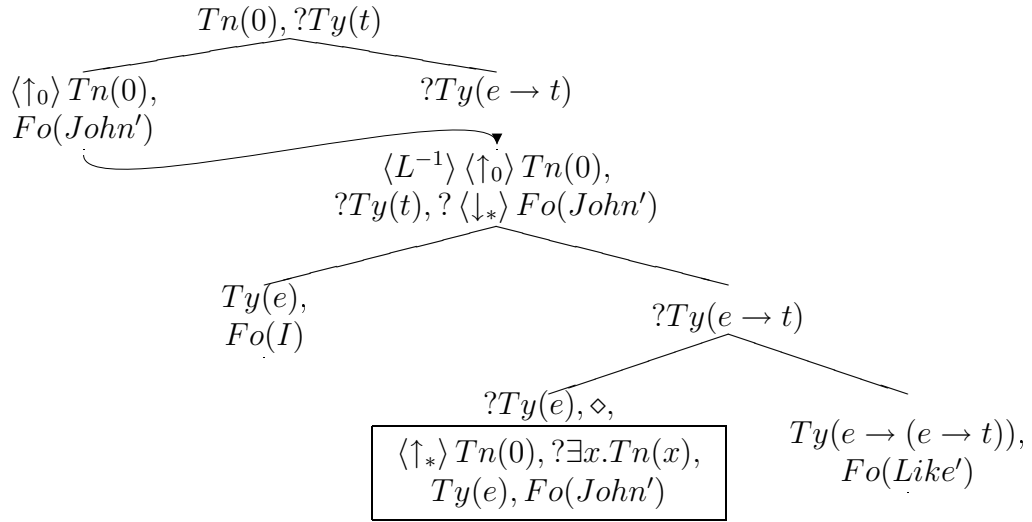
(21)	$who_{rel}$	IF	$?Ty(e), ?\exists x.Tn(x),$		
			$\langle \uparrow_* \rangle \langle L^{-1} \rangle Fo(y)$		
		THEN	IF	$\uparrow \top$	
			THEN	Abort	
		ELSE		$put(Fo(y), Ty(e), [\downarrow] \perp)$	
		ELSE	Abort		

Cette description lexicale précise que SI, en remontant du noeud non-fixe et en remontant la relation de Lien,  $Fo(y)$  est vrai, ALORS il faut mettre une copie à l'endroit où l'on est (là où le pronom relatif est parsé).

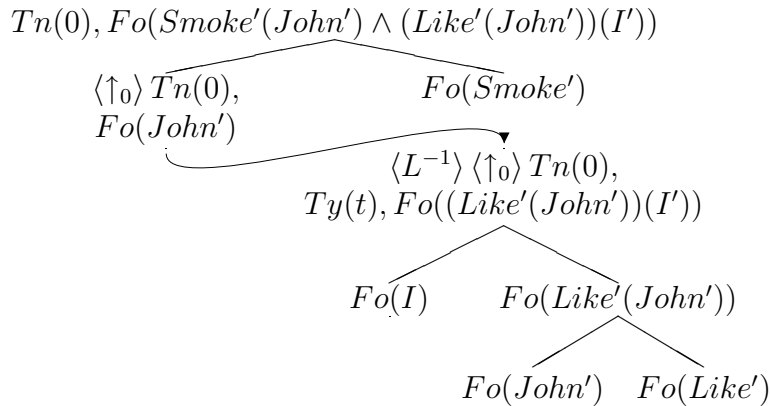
La commande  $\uparrow \top$  permet juste d'éviter de générer des phrases comme *\*John helped any student the work of who was inadequate*. Elle spécifie que *who* ne peut être inséré sur un noeud fixe. Par contre, ceci est possible avec *which* : sa description lexicale ne contiendra pas  $\uparrow \top$  pour permettre des phrases du type *This shirt, the collar of which was faded, was on sale*.



L'arbre lié va ensuite se développer comme une structure de dislocation (ex *:John, I like*), avec un noeud non-fixe qui sera unifié (Merge) avec la position d'objet de *like* :



Il suffit maintenant de compléter l'arbre lié (ôter les requêtes et accomplir les applications fonctionnelles) et de terminer la phrase principale (insertion de *smoke*) :



## 5.2 Le croisement

**Effet de croisement fort (en GG)** : dès qu'un constituant se déplace vers une position A' en croisant sur son chemin un élément anaphorique qui le c-commande et avec lequel il est coindicé, la phrase devient agrammaticale.

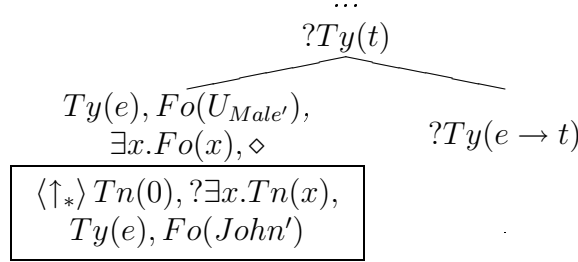
(22) John<sub>1</sub>, who<sub>1</sub> (I am certain) said he<sub>1</sub> would be at home, is in the surgery.

(23) \*John<sub>1</sub>, who<sub>1</sub> (I am certain) he<sub>1</sub> said would be at home, is in the surgery.

⇒ revient à un effet de reconstruction avec la condition C.

**Explication de l'agrammaticalité en (23)** : il s'agit d'une interaction entre deux types de sous-spécification, la sous-spécification des pronoms (Substitution) et la sous-spécification de noeud (Unification de noeud). Le pronom *he* en (23) va devoir récupérer sa référence par unification de noeud, et pas par Substitution (qui sera exclue par notre nouvelle condition B). Techniquement, comment ça marche ?

-Insertion *he* ⇒



La condition de localité ( $\langle \uparrow_0 \rangle \langle \uparrow_1^* \rangle \langle \downarrow_0 \rangle Fo(\alpha)$ ) va empêcher la Substitution car  $Fo(John')$  est déjà présent sur le même noeud que la métavariable  $U$ .

$$(24) \quad \text{Subst}(\alpha) \quad \left| \begin{array}{ll} \text{IF} & Fo(U), Ty(e) \\ \text{THEN} & \text{IF} \quad \langle \uparrow_0 \rangle \langle \uparrow_1^* \rangle \langle \downarrow_0 \rangle Fo(\alpha) \\ & \text{THEN} \quad \text{Abort} \\ & \text{ELSE} \quad \text{put}(Fo(\alpha)) \\ \text{ELSE} & \text{Abort} \end{array} \right.$$

⇒ Seule l'Unification des noeuds est possible, mais la fin de la phrase ne pourra être analysée car il n'y aura plus rien à unifier avec le sujet de *would be at home*.

**Cas problématique** : comment expliquer la grammaticalité de (25), cas de croisement fort étendu ?

(25) John<sub>1</sub>, whose<sub>1</sub> mother he<sub>1</sub> worries about, has stopped working.

Avec ce système, l'absence de croisement dans cet exemple est expliquée, car il n'y aura pas de concurrence entre Unification et Substitution :

-l'unification entre le noeud non-fixe de *whose mother* et le noeud fixe de *he* sera impossible.

-la Substitution sera possible car le noeud non-fixe contiendra une formule du type  $Fo(Mother'(John'))$ , la localité ne sera donc pas violée.





## 6 Résomptivité dans les relatives : typologie

**Rappel :** un pronom résomptif est un pronom de reprise (qui reprend une expression de la phrase) et qui occupe une position où une lacune est attendue.

(27) Le garçon dont je sais qu'**il** ne viendra pas.

Les pronoms résomptifs sont souvent distingués des pronoms classiques car ces derniers ont une référence assez libre dans le contexte, alors que la référence des pronoms résomptifs est contrainte (*il* ne peut renvoyer qu'au garçon).

Les différentes hypothèses de sous-spécification, ainsi que la manière dont sont dérivées les relatives, vont permettre de dégager une typologie de la résomptivité dans ces constructions : le fait que la résomptivité est plus productive dans certaines langues que d'autres.

Pour Kempson, il n'y a pas différents types de pronoms (classiques, intrusifs, résomptifs...), mais un seul :

he	IF      ? <i>Ty</i> ( <i>e</i> ) THEN put( <i>Ty</i> ( <i>e</i> ), <i>Fo</i> ( <i>U<sub>Male'</sub></i> ), ? $\exists x.Fo(x)$ , ? $\langle \uparrow \rangle (Ty(t) \wedge \exists y.Tns(y))$ , $[\downarrow \perp]$ ; ELSE ABORT	Motivation Type et Formule de présupposition Requête d'une formule Condition de cas Noeud Terminal
----	--	--

### 6.1 Résomptivité en Anglais

Anglais : langue où la stratégie résomptive ne semble pas être la stratégie par défaut, mais plutôt une stratégie secondaire ou de dernier recours dans des structures complexes (îles pour le mouvement).

(28) ?He's a professor who John liked him.

Même si les locuteurs rejettent (28) en faveur de *he's a professor who John liked*, il s'avèrent que de nombreux locuteurs la produisent.

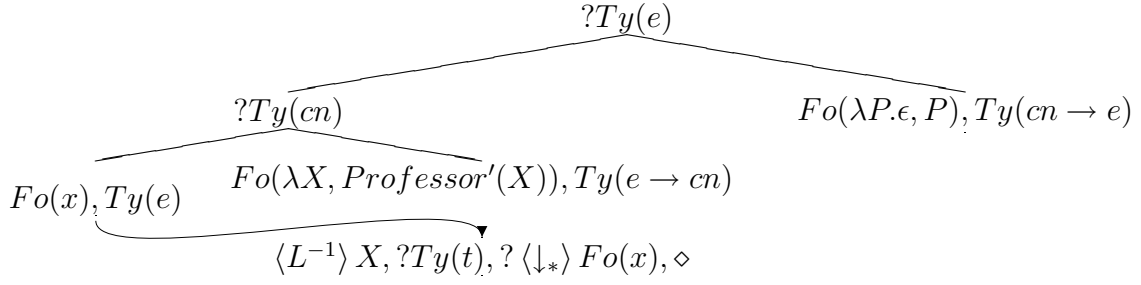
Exemple avec une île pour le mouvement :

(29) I have three people that I don't know how **they** behave in other classes.

La possibilité d'avoir des pronoms résomptifs en Anglais est prédite :

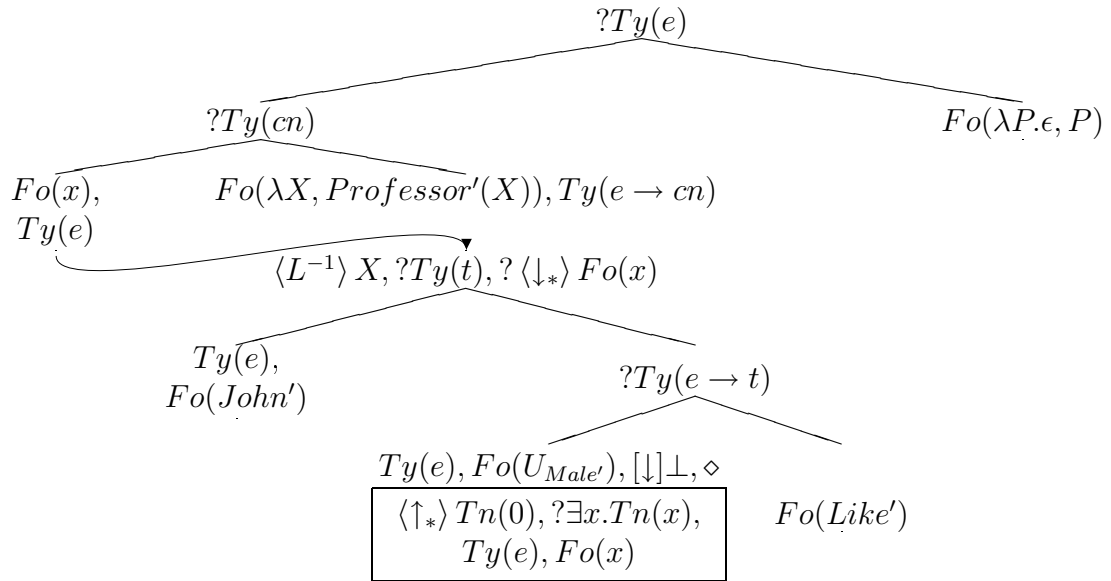
-Insertion *a professor* + Adjonction de lien :

⇒



-Adjonction+traitement de *who liked him* :

⇒



A ce moment de l'incrémentation, rien n'empêche d'unifier le pronom *him* avec le noeud non-fixe contenant la copie de  $Fo(x)$ . La 'bottom restriction' spécifie seulement qu'on a un noeud terminal (qui ne peut pas avoir de filles), mais l'unification est possible.

## 6.2 Résomptivité en Arabe Egyptien

Arabe égyptien : langue où les pronoms résomptifs sont obligatoires dans toutes les positions sauf le sujet (stratégie résomptive productive) et peuvent aussi apparaître dans des îles pour le mouvement.

- (30) *il mudarris illi Magdi darab-u.*  
 le professeur que Magdi a frappé-le  
 'Le professeur que Magdi a frappé'

Exemple avec une île :

- (31) *tʔarrafn*                      *ʔala l-muxrizh*    *yalli laila sheefit l-masrahiyye*  
 nous avons rencontré sur le directeur que laila a vu la pièce  
*yalli huwwe ʔaxrazh-a.*  
 que il a dirigé-la  
 ‘Nous avons rencontré le directeur dont Laila a vu la pièce qu’il a dirigée.’

Il est très simple d’expliquer la présence obligatoire d’un pronom résomptif en Arabe Egyptien. Il suffit de spécifier que le complémenteur de relativisation ne fait que créer une structure liée sans fournir une copie de l’antécédent (à la différence de *who* en Anglais qui crée une copie de ‘la formule relativisée’). Autrement dit, on n’a pas de relatives avec un noeud non-fixe en Arabe égyptien :

$$illi \left\{ \begin{array}{ll} \text{IF} & Fo(x), Ty(e), [\downarrow] \perp \\ \text{THEN} & \text{make}(\langle L \rangle); \text{go}(L); \text{put}(\text{?}Ty(t), \text{?} \langle D \rangle (Fo(x), Ty(e))) \\ \text{ELSE} & \text{ABORT} \end{array} \right.$$

Pour (30), le traitement du complémenteur impose une structure liée avec une requête pour une copie de  $Fo(x)$ , mais pas de noeud non-fixe. Quand le pointeur atteint l’objet du verbe, aucune unification n’est possible car il n’y a pas de noeud non-fixe. La seule option est d’avoir un pronom (résomptif) avec la formule  $Fo(U)$ . Cette formule va enfin pouvoir être modifiée par Substitution pour devenir  $Fo(x)$  <sup>2</sup>.

Le fait que les pronoms résomptifs peuvent apparaître dans des îles en Arabe égyptien est spécifié grâce à l’opérateur  $\langle D \rangle$  qui spécifie que la copie peut apparaître non seulement dans des relations de filles, mais aussi à travers les relations de lien :

-?  $\langle \downarrow_* \rangle Fo(\alpha)$      $\Rightarrow$  n’importe où dans l’arbre (lié) que je domine ;  
 -?  $\langle D \rangle Fo(\alpha)$      $\Rightarrow$  n’importe où dans les arbres (toutes les structures liées) que je domine.

### 6.3 Résomptivité en Roumain

Roumain : langue où les clitiques résomptifs sont obligatoires mais ne peuvent apparaître dans les îles pour le mouvement.

- (32) *băiatul pe care l-am văzut.*  
 le garçon ?? que le-j’ai vu  
 ‘le garçon que j’ai vu.’

Ce pronom résomptif ne peut apparaître dans les îles :

- (33) *\*Omul pe care cunosc femeia care l-a întâlnit.*  
 l’homme ? que (je) connais la femme qui le-a rencontré  
 ‘l’homme dont je connais la femme qui l’a rencontré.’

---

<sup>2</sup>Pour la position sujet, l’insertion du pronom résomptif n’est pas obligatoire car le verbe fournit déjà une formule pour le sujet.

Le complémenteur du Roumain va donc se comporter comme celui de l'Arabe égyptien, sauf que la copie devra être fournie dans l'arbre lié. Dans l'information lexicale de *care*, il faut donc remplacer la requête  $? \langle D \rangle (Fo(\alpha))$  par la requête  $? \langle \downarrow_* \rangle (Fo(\alpha))$ .

## 6.4 Résomptivité en Hébreu

Hébreu : langue où le choix est assez libre entre stratégie résomptive et construction avec une lacune.

- (34) *ze ha-ʔiš še (ʔoto) raʔiti ʔetmoi.*  
 c'est l'homme que lui je vis hier  
 'C'est l'homme que j'ai vu hier.' *Demirdache (1992)*

Pour expliquer cette liberté, il suffit de dire que l'Hébreu est un mélange d'Arabe égyptien et d'Anglais. Cette langue ressemble à l'Arabe car le complémenteur n'insère que la requête faible  $? \langle D \rangle Fo(x)$ . Elle va ressembler à l'Anglais car l'Adjonction sera possible, créant un noeud non-fixe qui peut être décoré par le pronom résomptif. Cette analyse prédit que le résomptif apparaîtra à la périphérie gauche des noeuds  $Ty(t)$ , ce qui est confirmé par les exemples suivants :

- (35) *ha-ʔ-iš še ʔalav ʔani xošev še ʔamarta še sara katva*  
 l'homme que sur lui je pense que tu as dit que Sara a écrit  
*šir*  
 un poème  
 'L'homme sur qui je pense que tu as dit que Sara a écrit un poème.'

- (36) *ha-ʔ-iš še ʔani xošev še ʔalav ʔamarta še sara katva*  
 l'homme que je pense que sur lui tu as dit que Sara a écrit  
*šir*  
 un poème  
 'L'homme sur qui je pense que tu as dit que Sara a écrit un poème.'

- (37) *ha-ʔ-iš še ʔani xošev še ʔamarta še ʔalav sara katva*  
 l'homme que je pense que tu as dit que sur lui Sara a écrit  
*šir*  
 un poème  
 'L'homme sur qui je pense que tu as dit que Sara a écrit un poème.'

## 7 La périphérie gauche : typologie

Nous avons déjà vu une manière de traiter des phrases avec un élément en périphérie gauche : la notion de noeud non-fixe.

→ utilisé pour la topicalisation en Anglais, et pour le sujet marqué dans les langues

pro-drop.

La notion de lien (LINK) va offrir une autre manière de construire de la dislocation à gauche. En effet, la notion de lien utilisée pour l'adjonction (lien inverse  $\langle L^{-1} \rangle$  de type  $?Ty(e)$  vers  $?Ty(t)$ ) va être étendue à la notion de dislocation à gauche (lien de type  $?Ty(t)$  vers  $?Ty(e)$ ) car les deux phénomènes montrent certains points communs. On aboutit à la règle suivante :

$$\text{Introduction de dislocation : } \frac{\{\{Tn(0), ?Ty(t), \diamond\}\}}{\{\{Tn(0), ?Ty(t)\}, \{\langle L \rangle Tn(0), ?Ty(e), \diamond\}\}}$$

Cette règle va permettre de créer la structure suivante :

