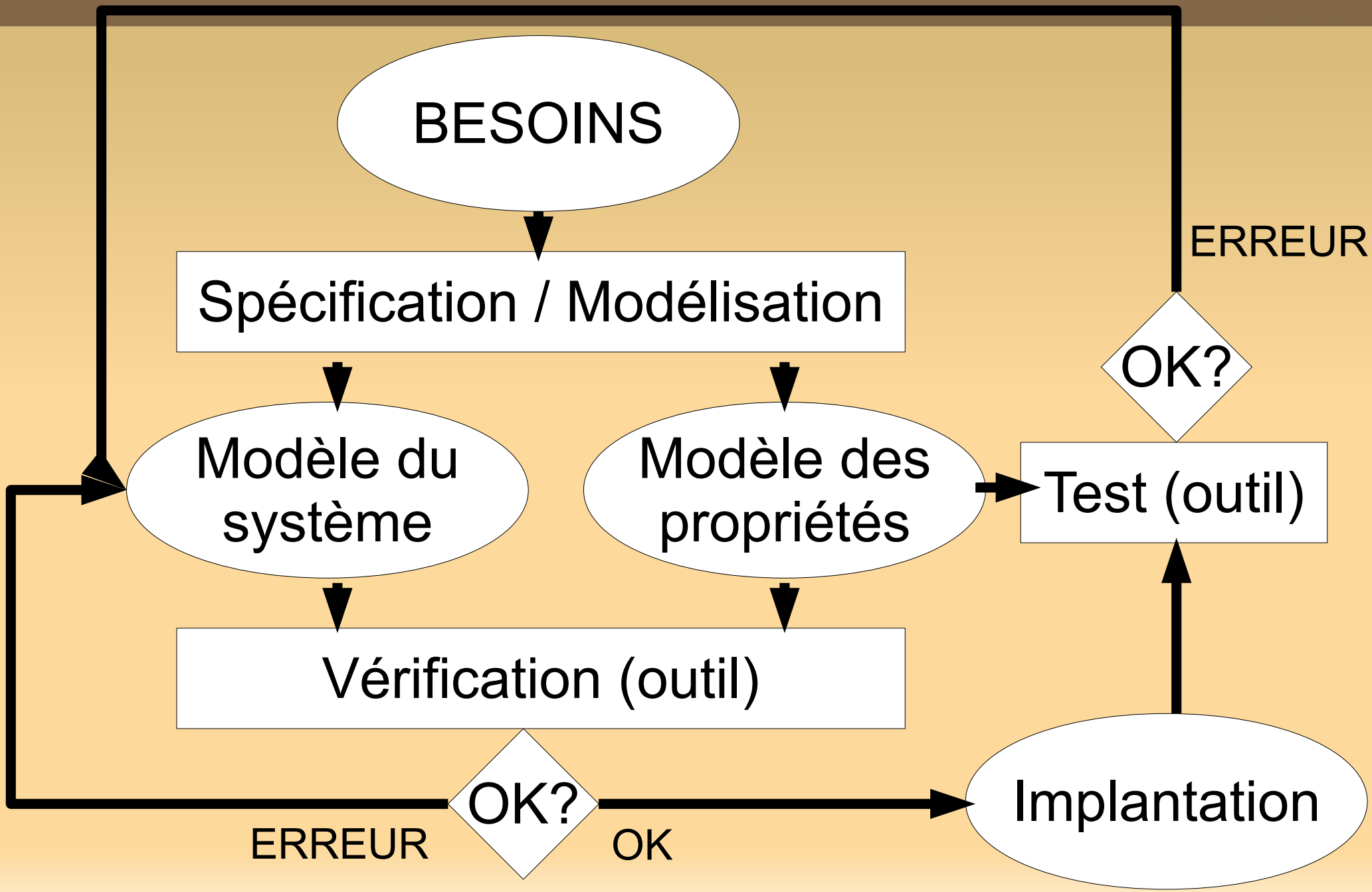
Partie 3 – Logique temporelle description des propriétés

- Partie 1 – modèle comportemental de système
- Partie 2 – ”langages” de processus
- Partie 3 – propriétés

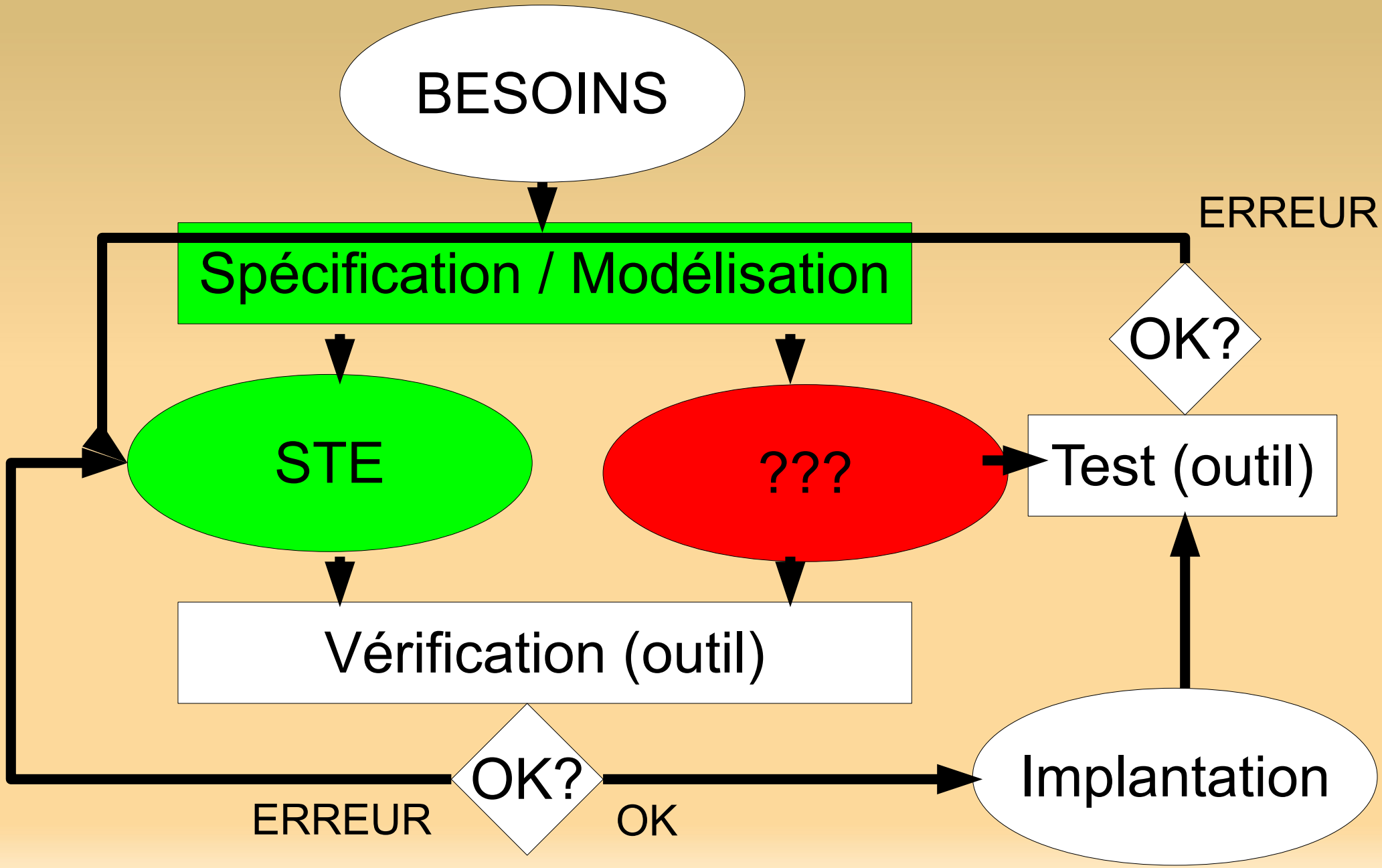
Retour sur la méthode ...



Le point et les objectifs

- un système S peut être modélisé à l'aide de LTS
  qu'il soit simple ou composite / communicant ou non

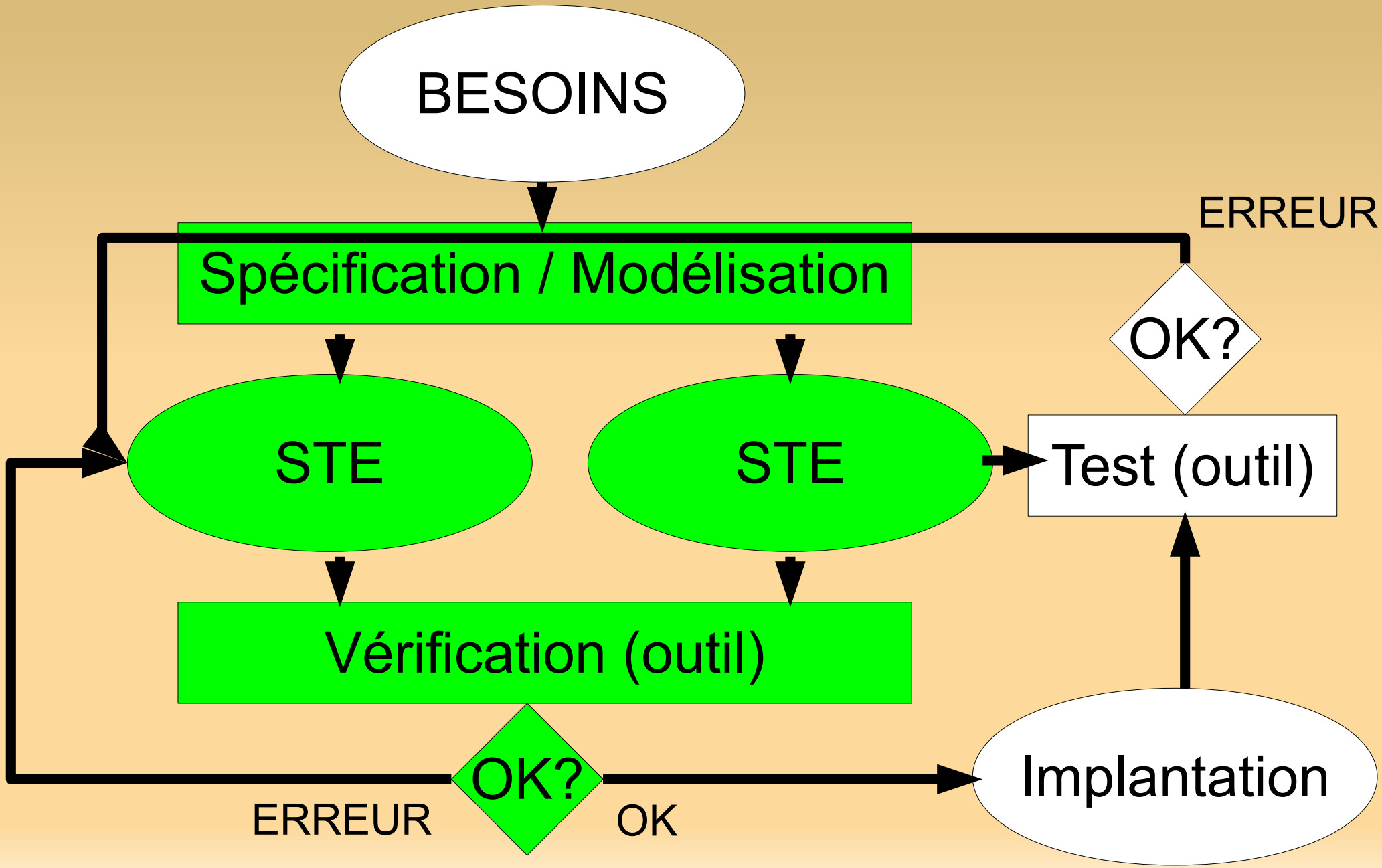
Retour sur la méthode ...



Le point et les objectifs

- un système S peut être modélisé à l'aide de STE
  qu'il soit simple ou composite / communicant ou non
- on a vu comment vérifier des choses simples:
  - absence de blocage: recherche des états puits
  - vivacité/possibilité de faire qqch: trace t $\in$ traces(S)
  - correspondance à un besoin (STE B): SRB, R$\in\{\sim,\approx\}$

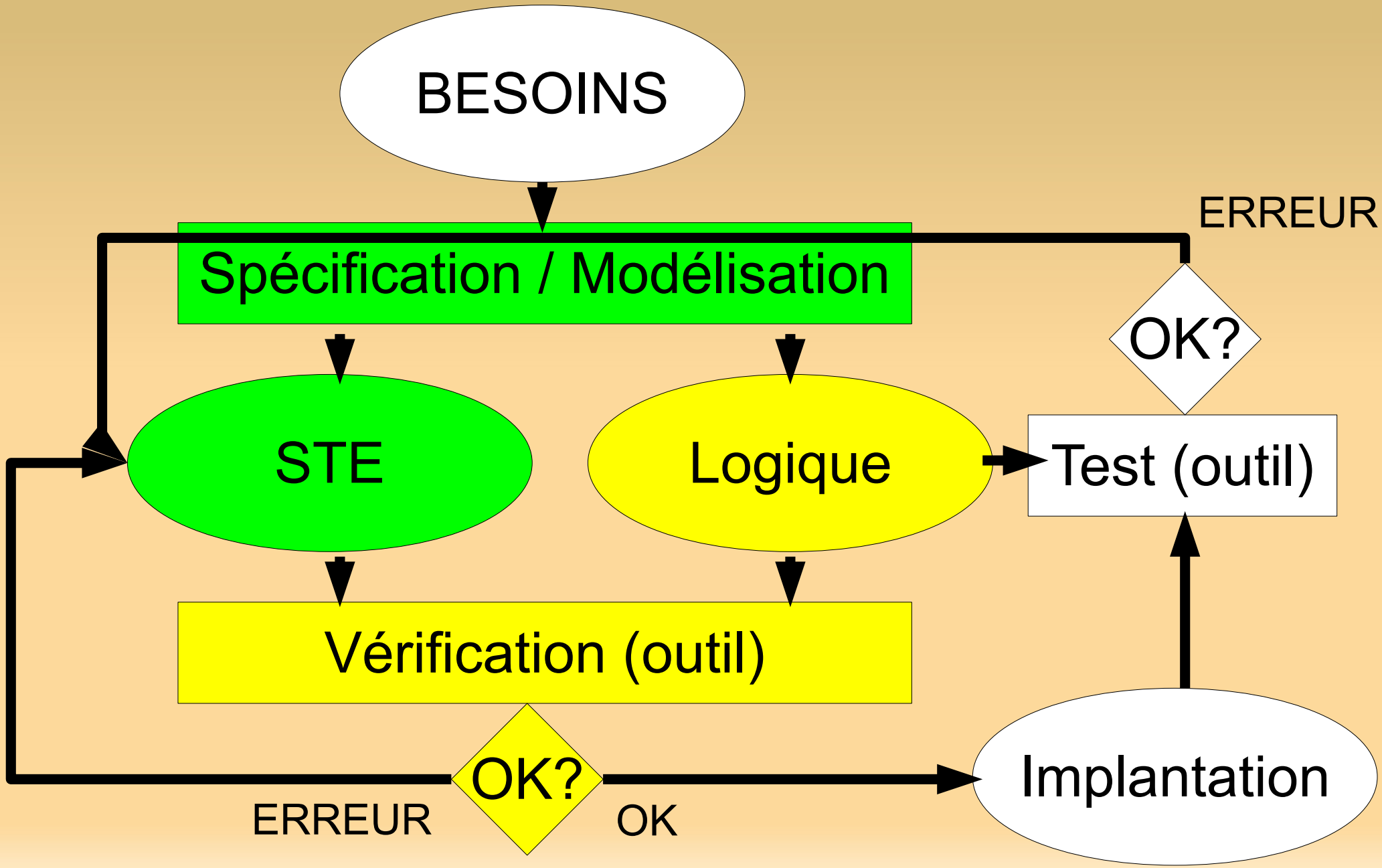
Retour sur la méthode ...



Le point et les objectifs

- un système S peut être modélisé à l'aide de STE
  qu'il soit simple ou composite / communicant ou non
- on a vu comment vérifier des choses simples:
  - absence de blocage: recherche des états puits
  - vivacité/possibilité de faire qqch: trace t $\in$ traces(S)
  - correspondance à un besoin (STE B): SRB, R$\in\{\sim,\approx\}$
- comment définir les propriétés attendues
  - de façon plus générale ?
  - de façon plus abstraite ?
- -> une logique spécifique

Retour sur la méthode ...



Rappels de logique

- un système logique (ou logique) est défini par
  - une syntaxe
    définit des symboles et des règles pour les combiner
    → formules (bien formées)
  - un système de déduction (ou calcul)
    permet de raisonner syntaxiquement
    → preuves, formules prouvables / théorèmes, |-
  - une sémantique / fonction d'interprétation
    permet d'associer à toute formule un objet dans une structure abstraite appelée modèle
    → validité de formules, vérification de modèle, |=

Rappels de logique

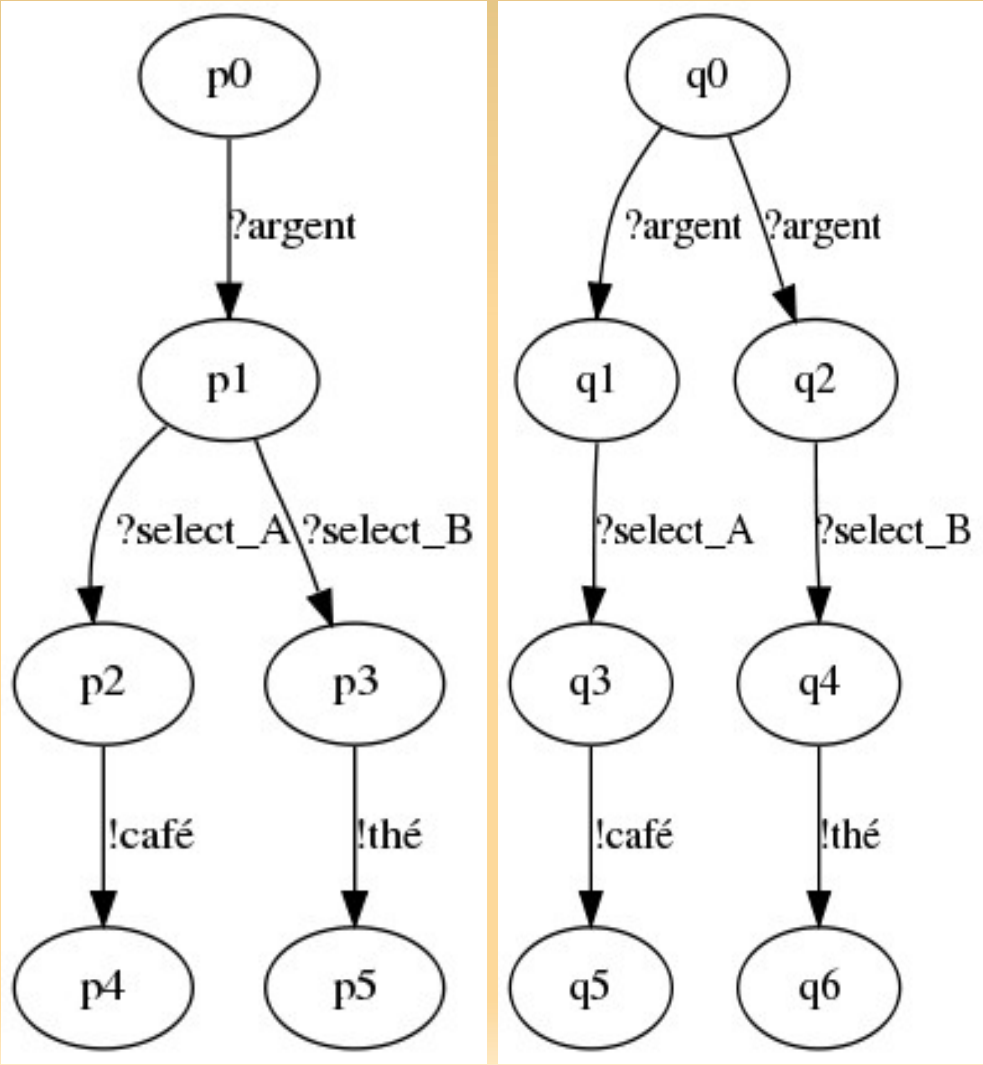
- un système logique (ou logique) est défini par
  - une syntaxe
  - un système de déduction (ou calcul), [...], |-
  - une sémantique / fonction d'interprétation, [...], |=
- un calcul est
  - correct si pour toute formule Φ, |-Φ => |=Φ
  - complet si pour toute formule Φ, |=Φ => |-Φ
- une formule de logique est
  - satisfiable s'il existe un modèle qui la vérifie
  - valide si tout modèle la vérifie

Vers une première logique ...

- $\Phi ::= tt \mid \neg\Phi \mid \Phi 1 \vee \Phi 2$
- de plus :
  - $ff == \neg tt$
  - $\neg\Phi 1 \wedge \neg\Phi 2 == \neg(\Phi 1 \vee \Phi 2)$
  - $\Phi 1 \Rightarrow \Phi 2 == (\neg\Phi 1) \vee \Phi 2$
  - $\Phi 1 \Leftrightarrow \Phi 2 == (\Phi 1 \Rightarrow \Phi 2) \wedge (\Phi 2 \Rightarrow \Phi 1)$
- que nous manque-t-il
  par ex., p/r à la logique des propositions ?
  $\longrightarrow$ les ”atomes propositionnels”
  $\longrightarrow$ ici (pour nos systèmes/STE) : ?

Vers une première logique ...

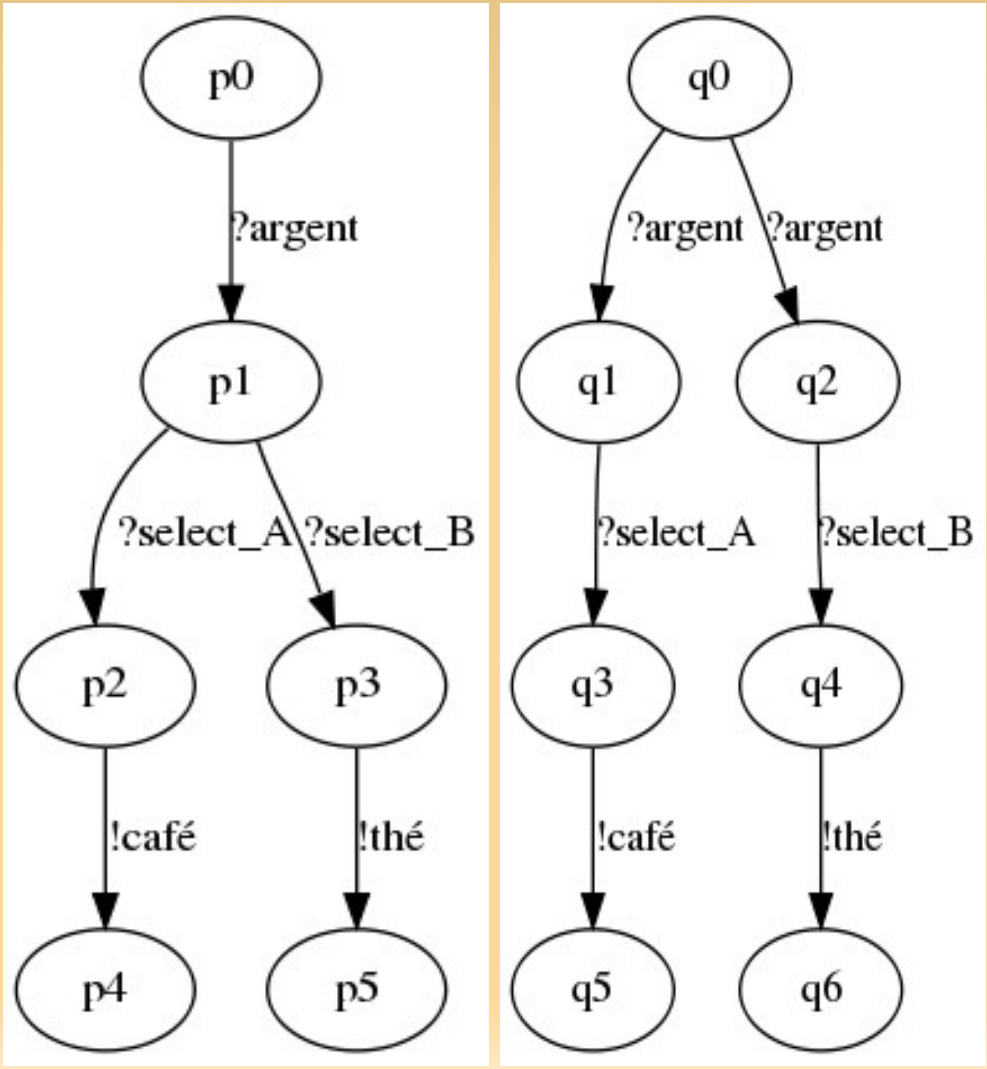
- revenons sur nos deux machines, M1 et M2:
- rappel: M1≁M2
- en quoi se différencient-elles ?



Vers une première logique ...

- revenons sur nos deux machines, M1 et M2:
- rappel: M1 $\not\sim$ M2
- en quoi se différencient-elles ?

  M1: après ?argent
  on peut toujours avoir ?select_A
  M2: il existe un ?argent
  après lequel on peut
  ne pas pouvoir avoir ?select_A



HML

- Hennessy-Milner Logic
- $\Phi_{HML} ::= tt \mid \neg\Phi \mid \Phi1 \vee \Phi2 \mid \langle a \rangle \Phi$
- dérivés
  - $ff == \neg tt$
  - $\neg\Phi1 \wedge \neg\Phi2 == \neg(\Phi1 \vee \Phi2)$
  - $\Phi1 \Rightarrow \Phi2 == (\neg\Phi1) \vee \Phi2$
  - $\Phi1 \Leftrightarrow \Phi2 == (\Phi1 \Rightarrow \Phi2) \wedge (\Phi2 \Rightarrow \Phi1)$
  - $[a]\Phi == \neg\langle a \rangle\neg\Phi$

POSSIBILITE
il est possible de faire a et passer dans un état qui vérifie Φ

NECESSITE
tous les a font passer dans un état qui vérifie Φ

HML

- Hennessy-Milner Logic
- $\Phi_{HML} ::= tt \mid \neg\Phi \mid \Phi 1 \vee \Phi 2 \mid \langle a \rangle \Phi$
- dérivés
  - $ff == \neg tt$
  - $\neg\Phi 1 \wedge \neg\Phi 2 == \neg(\Phi 1 \vee \Phi 2)$
  - $\Phi 1 \Rightarrow \Phi 2 == (\neg\Phi 1) \vee \Phi 2$
  - $\Phi 1 \Leftrightarrow \Phi 2 == (\Phi 1 \Rightarrow \Phi 2) \wedge (\Phi 2 \Rightarrow \Phi 1)$
  - $[a]\Phi == \neg\langle a \rangle \neg\Phi$

M1:
après ?argent
on peut toujours
avoir ?select_A
→ ???

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
→ ???

HML

- Hennessy-Milner Logic
- Φ_{HML} ::= tt | ¬Φ | Φ1∨Φ2 | <a>Φ
- dérivés
  - ff == ¬tt
  - ¬Φ1∧¬Φ2 == ¬(Φ1∨Φ2)
  - Φ1⇒Φ2 == (¬Φ1)∨Φ2
  - Φ1⇔Φ2 == (Φ1⇒Φ2)∧(Φ2⇒Φ1)
  - [a]Φ == ¬<a>¬Φ

M1:
après ?argent
on peut toujours
avoir ?select_A
[?argent]<?select_A>tt

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
<?argent>[?select_A]ff

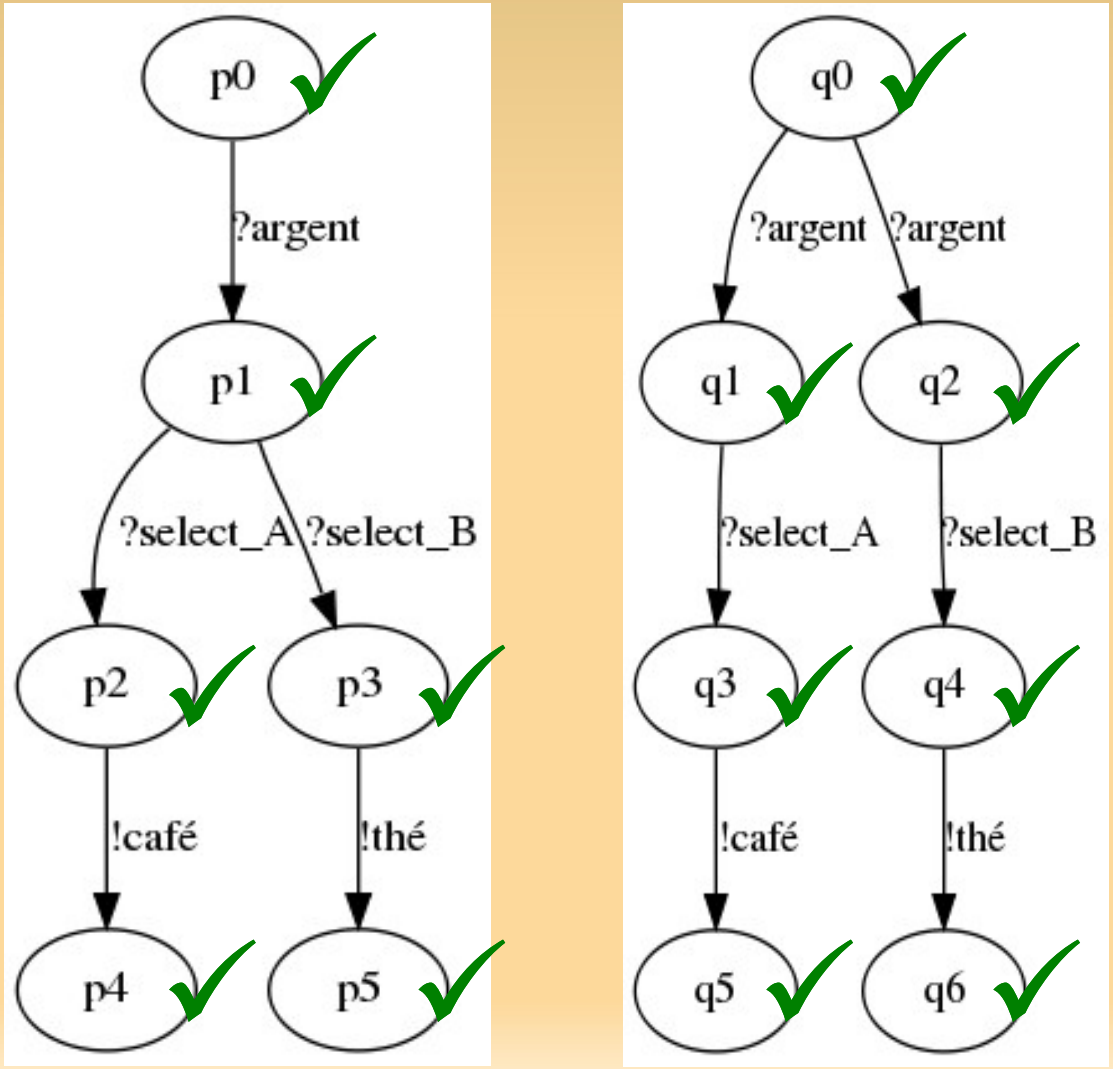
Sémantique de HML

- interprétation pour un STE (A,S,I,F,T): $||\Phi|| \subseteq S$
  - $||tt|| = S$
  - $||\neg\Phi|| = S \setminus ||\Phi||$ $||ff|| = \emptyset$
  - $||\Phi1 \vee \Phi2|| = ||\Phi1|| \cup ||\Phi2||$ $||\Phi1 \wedge \Phi2|| = ||\Phi1|| \cap ||\Phi2||$
  - $||\langle a \rangle \Phi|| = \{s \in S | \exists\, s' \in S : (s,a,s') \in T \wedge s' \in ||\Phi||\}$
  - $||[a]\Phi|| = \{s \in S | \forall s' \in S: (s,a,s') \in T \Rightarrow s' \in ||\Phi||\}$
- $L \models_s \Phi$ (aussi noté $s \models \Phi$) ssi $s \in ||\Phi||$

  L valide Φ au niveau de l'état s ssi s valide Φ
- $L \models \Phi$ ssi $L \models_{s0} \Phi$

Exemple

[?argent]<?select_A>tt

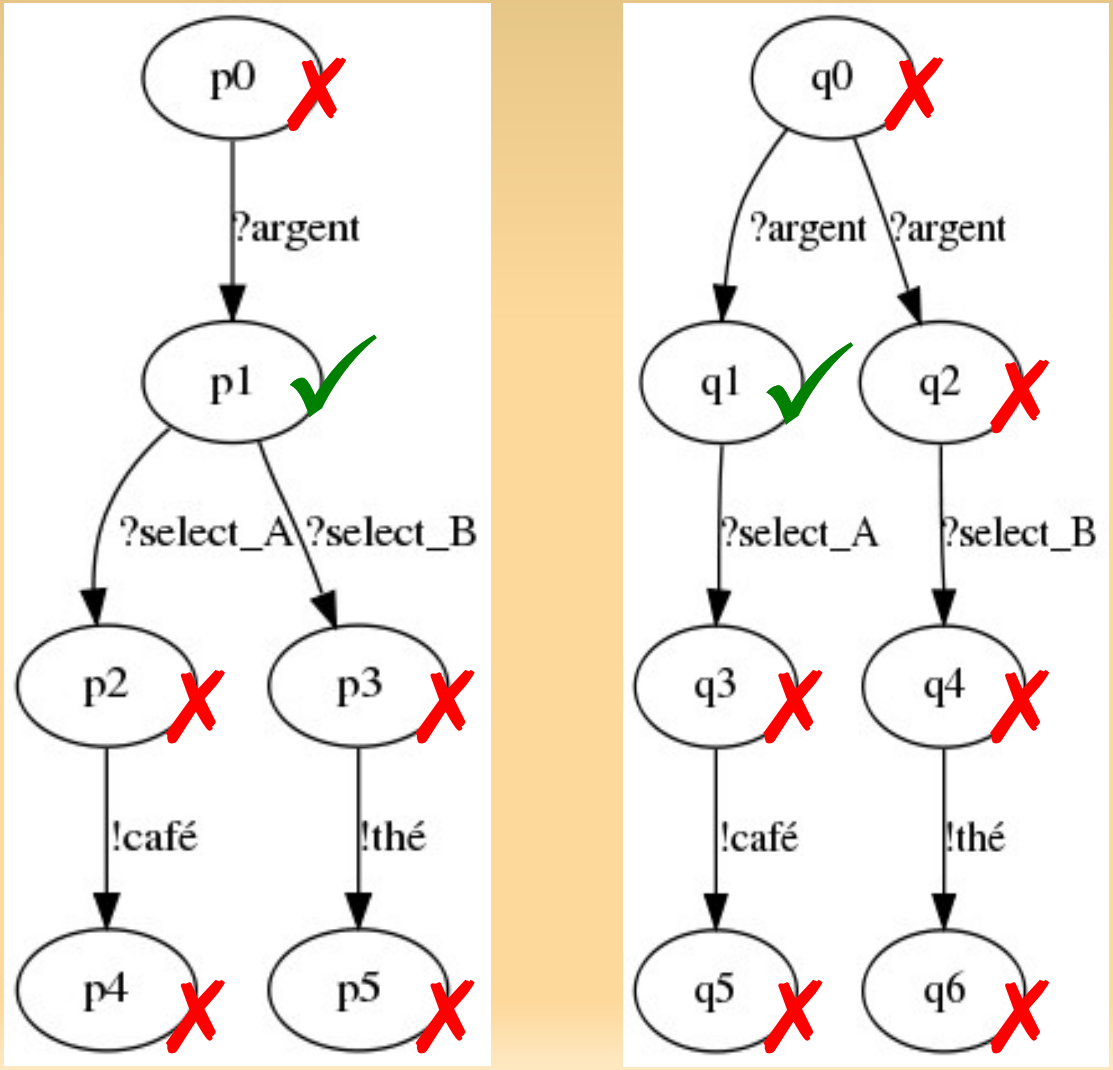


M1:
après ?argent
on peut toujours
avoir ?select_A
[?argent]<?select_A>tt

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
<?argent>[?select_A]ff

Exemple

[?argent]<?select_A>tt

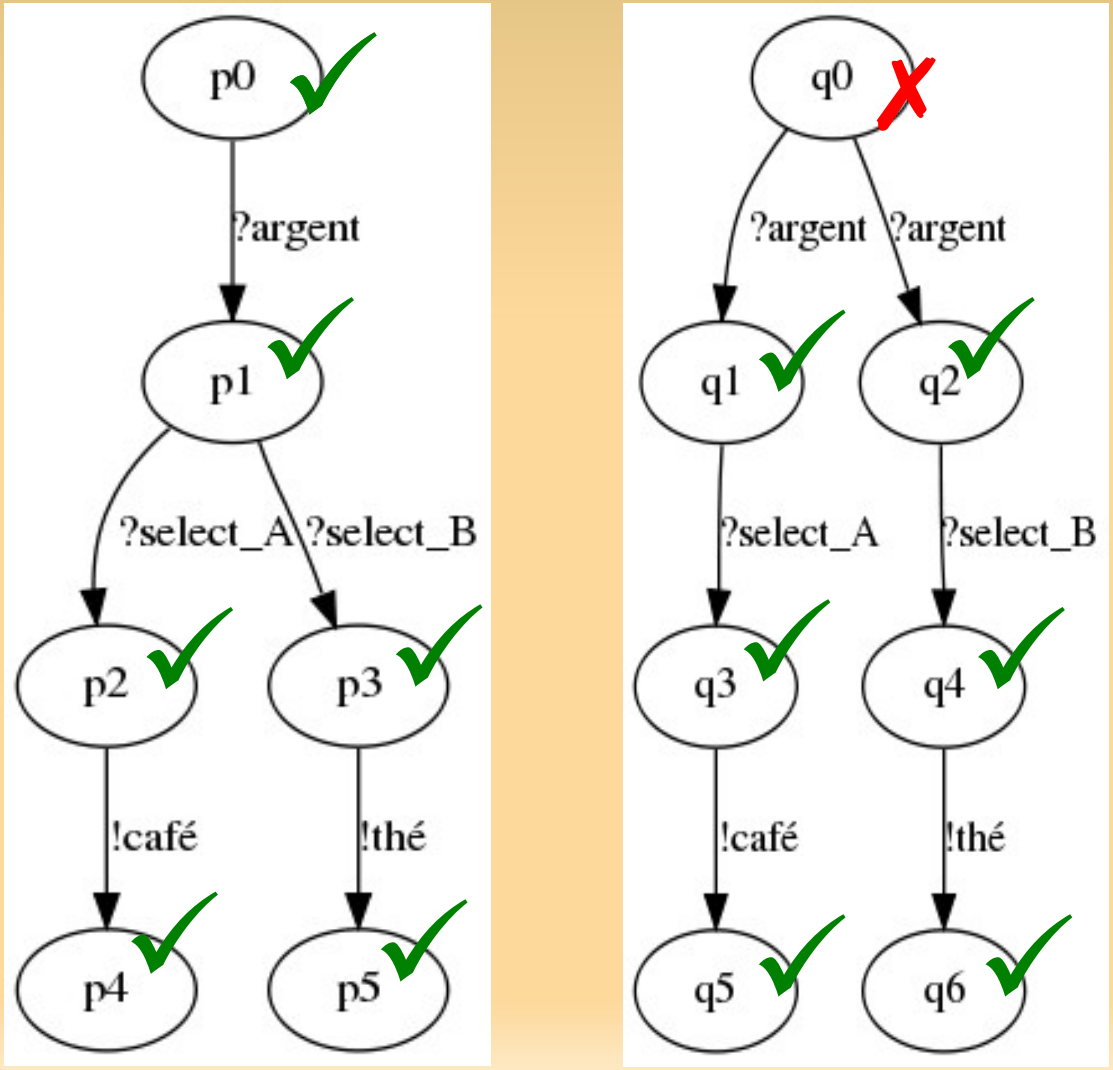


M1:
après ?argent
on peut toujours
avoir ?select_A
[?argent]<?select_A>tt

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
<?argent>[?select_A]ff

Exemple

[?argent]<?select_A>tt

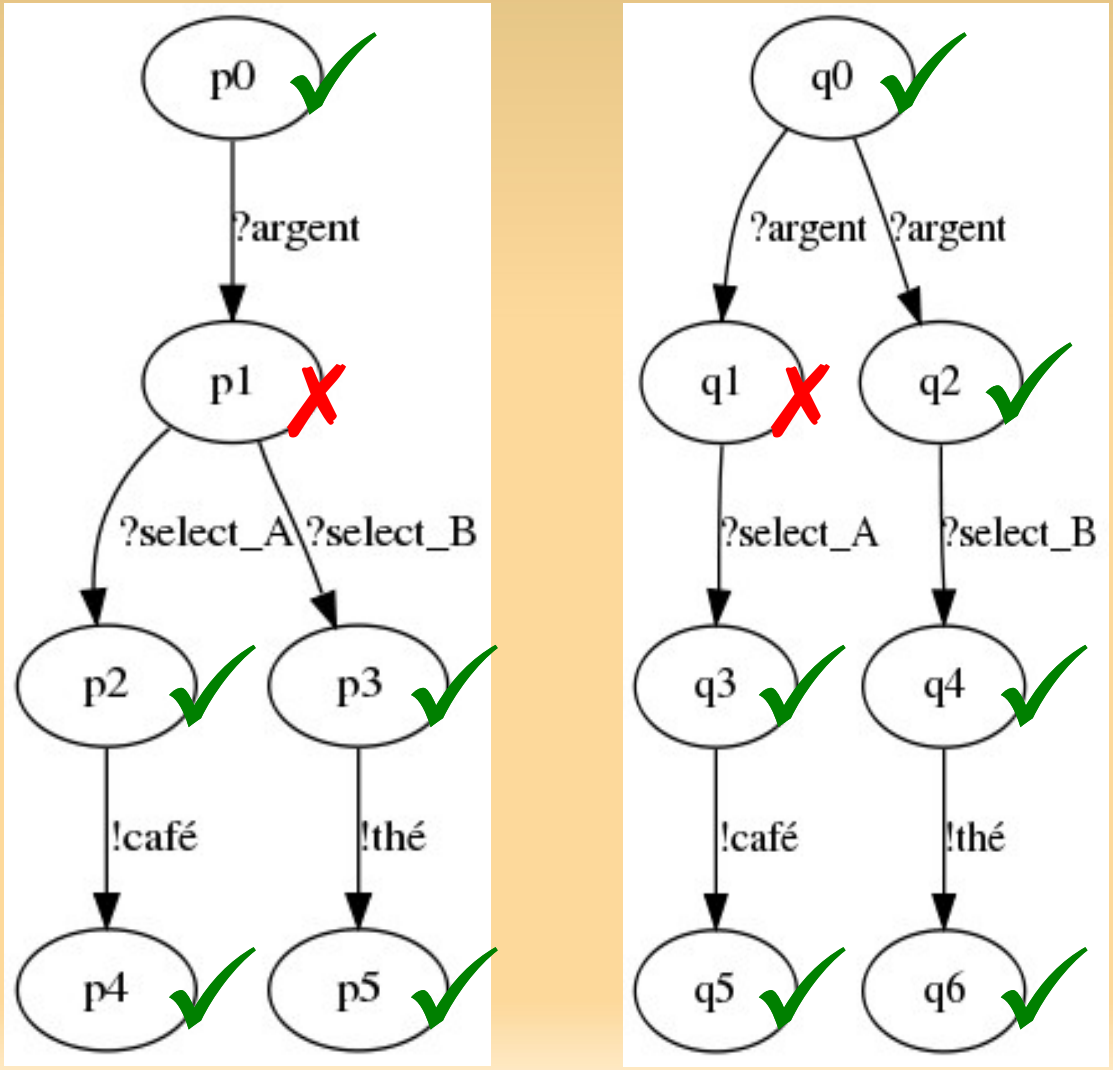


M1:
après ?argent
on peut toujours
avoir ?select_A
[?argent]<?select_A>tt

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
<?argent>[?select_A]ff

Exemple

<?argent>[?select_A]ff

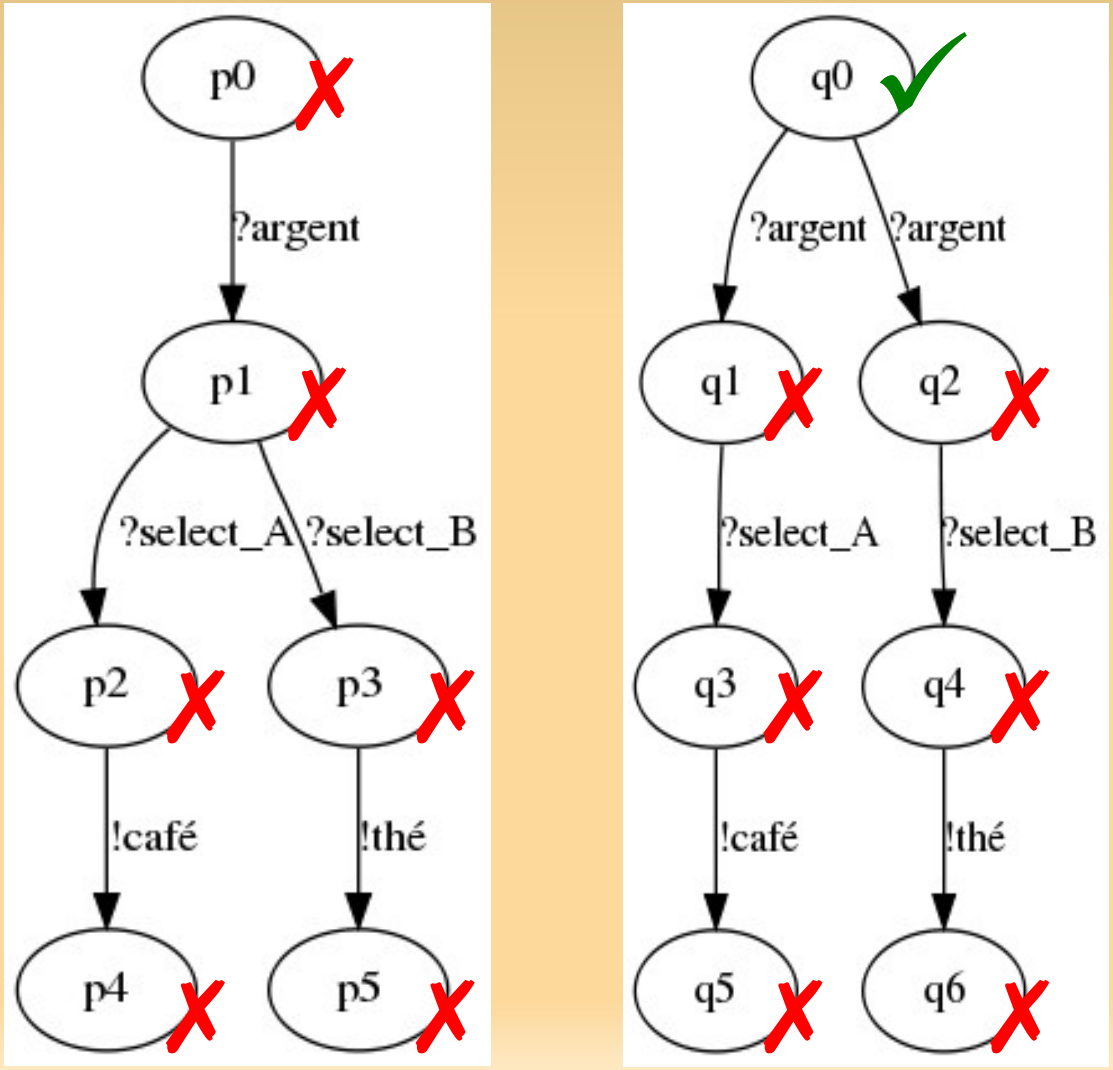


M1:
après ?argent
on peut toujours
avoir ?select_A
[?argent]<?select_A>tt

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
<?argent>[?select_A]ff

Exemple

<?argent>[?select_A]ff



M1:
après ?argent
on peut toujours
avoir ?select_A
[?argent]<?select_A>tt

M2:
il existe un ?argent
après lequel on peut
ne pas pouvoir
avoir ?select_A
<?argent>[?select_A]ff

Formules d'actions

- on peut autoriser des formules d'actions, α
- Φ_{HML} ::= tt | ¬Φ | Φ1∨Φ2 | <α>Φ (plus les dérivés)
- α ::= tt | ¬α | α1∨α2 | a (plus les dérivés)
  - interprétation pour un STE (A,S,I,F,T): ||α|| ⊆ A
  - ||tt|| = A ||a|| = {a}
  - ||¬α|| = A\||α|| ||ff|| = ∅
  - ||α1∨α2 || = ||α1||∪||α2|| ||α1∧α2 || = ||α1||∩||α2||

||α1⇒α2|| = (A\||α1||)∪||α2||

||α1⇔α2|| = ((A\||α1||)∪||α2||) ∩ ((A\||α2||)∪||α1||)

Sémantique de HML (avec actions)

- interprétation pour un STE (A,S,I,F,T): $||\Phi|| \subseteq S$
  - $||tt|| = S$
  - $||\neg\Phi|| = S \setminus ||\Phi||$ $||ff|| = \emptyset$
  - $||\Phi1 \vee \Phi2|| = ||\Phi1|| \cup ||\Phi2||$ $||\Phi1 \wedge \Phi2|| = ||\Phi1|| \cap ||\Phi2||$
  - $||<\alpha>\Phi|| = \{s \in S | \exists\, s' \in S, a \in A: (s,a,s') \in T \wedge a \in ||\alpha|| \wedge s' \in ||\Phi||\}$
  - $||[\alpha]\Phi|| = \{s \in S | \forall\, s' \in S, a \in A: (s,a,s') \in T \wedge a \in ||\alpha|| \Rightarrow s' \in ||\Phi||\}$
- $L \models_s \Phi$ (aussi noté $s \models \Phi$) ssi $s \in ||\Phi||$

  L valide Φ au niveau de l'état s / s valide Φ
- $L \models \Phi$ ssi $L \models_{s0} \Phi$

Exercice HML

- écrire en HML:
  - il est possible de faire a
  - il est impossible de faire a
  - état de blocage (deadlock)
  - on ne peut faire que a ou b
  - il est impossible de pouvoir faire a et b
  - faire a rend impossible de faire b
  - on doit forcément faire a

Exercice HML

- écrire en HML:
  - il est possible de faire a — $\langle a \rangle tt$
  - il est impossible de faire a — $[a]ff$
  - état de blocage (deadlock) — $[tt]ff$
  - on ne peut faire que a ou b — $[\neg(a \vee b)]ff$
  - il est impossible de pouvoir faire a et b

    $\neg(\langle a \rangle tt \wedge \langle b \rangle tt)$ ou $[a]ff \vee [b]ff$
  - faire a rend impossible de faire b

    $[a][b]ff$ i.e $\langle a \rangle tt \Rightarrow [a][b]ff$
  - on doit forcément faire a

    $\langle a \rangle tt \wedge \neg\langle \neg a \rangle tt$ i.e $\langle a \rangle tt \wedge [\neg a]ff$

HML et ~

- La logique HML est adéquate p/r à la bissimulation forte (~)
- $L1 \sim L2 \Leftrightarrow (\forall \Phi \in \Phi_{HML}, L1 \models \Phi \Leftrightarrow L2 \models \Phi)$
- Donner une formule HML qui différencie L1 et L2 permet de prouver $L1 \not\sim L2$

wHML et ≈

- on a, de même, une logique adéquate pour ≈
- la logique weak HML est adéquate p/r à la bissimulation faible (≈)
- $L1 \approx L2 \Leftrightarrow (\forall \Phi \in \Phi_{wHML}, L1 \models \Phi \Leftrightarrow L2 \models \Phi)$
- Donner une formule wHML qui différencie L1 et L2 permet de prouver $L1 \not\approx L2$

wHML

- weak Hennessy-Milner Logic
- Φ_{wHML} ::= tt | ¬Φ | Φ1∨Φ2 | <<α>>Φ (plus les dérivés)
- || <<α>> Φ || = {s∈S|∃ s'∈S, a∈A : s=obs(a)=>s'∧ a∈|| α||∧s'∈||Φ||}
- |||[α]] Φ|| = {s∈S|∀s'∈S, a∈A : s=obs(a)=>s'∧ a∈||α|| ⇒s'∈ ||Φ||}
- inchangé p/r HML pour les autres

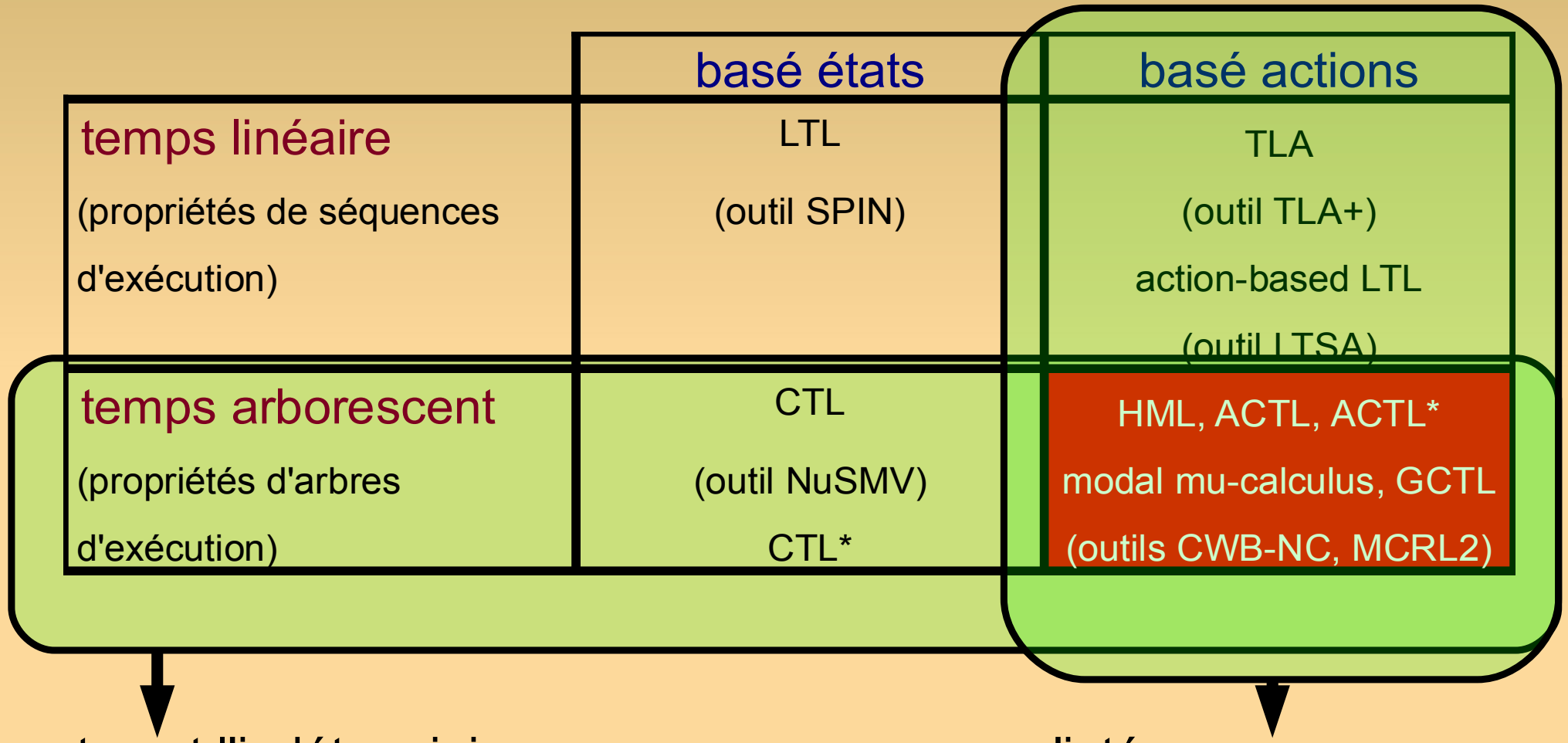
Vers une logique plus expressive

- (w)HML permettent d'exprimer des propriétés utiles et sont adéquates p/r ~ (≈)
- problème: (w)HML manquent d'expressivité pour exprimer des propriétés ”infinies” ou ”récursives”, type:
  - invariants/sureté: il est toujours possible de faire a, il n'est jamais possible de faire b, etc.
  - vivacité: il sera un jour possible de faire a
  - composés: à chaque fois qu'il est possible de faire a alors il est un jour possible de faire b

Vers une logique plus expressive

- (w)HML permettent d'exprimer des propriétés utiles et sont adéquates p/r à ~ (≈)
- problème: (w)HML manquent d'expressivité pour exprimer des propriétés ”infinies” ou ”récursives”, type [...]
  exemples:
  - il est toujours possible de faire a:
    <a>tt ∧ [tt]<a>tt ∧ [tt][tt]<a>tt ∧ ... ☹
  - il sera un jour possible de faire a:
    <a>tt ∨ <tt><a>tt ∨ <tt><tt><a>tt ∨ ... ☹

Logique linéaire ou arborescente ?



capturent l'indéterminisme
distinguent nos 2 machines à café
(mêmes séquences/traces mais pas ~)

on s'intéresse aux
observations (étiquettes des
transitions) et non aux états

R. Mateescu, Model Checking of Action-Based Concurrent Systems, VTSA'08.

HML + regform : intuition

- HML sans formules d'actions :
  - <a>Φ
    il existe une transition a
    qui ammène dans un état qui vérifie Φ
  - [a]Φ
    chaque transition a
    ammène dans un état qui vérifie Φ
- HML avec formules d'actions :
  - < α >Φ
    il existe une transition qui vérifie α
    et qui ammène dans un état qui vérifie Φ
  - [α]Φ
    chaque transition qui vérifie α
    ammène dans un état qui vérifie Φ

HML + regform : intuition

- pourquoi ne pas autoriser une expression dénotant une trace dans α ?
  - $\langle a.b.c \rangle \Phi$
    il existe une séquence de transitions a, b et c
    qui ammène dans un état qui vérifie Φ
  - $[a.b.c]\Phi$
    chaque séquence possible de transitions a, b et c
    ammène dans un état qui vérifie Φ
- et en allant plus loin, pourquoi ne pas pouvoir avoir :
  - 1 ou + : $\langle a^{+} \rangle \Phi$, $[a^{+}]\Phi$
  - 0 ou + : $\langle a^{*} \rangle \Phi$, $[a^{*}]\Phi$
  - formules régulières et formules d'action :
    $\langle \alpha.\alpha \rangle \Phi$, $[\alpha.\alpha]\Phi$, $\langle \alpha^{+} \rangle \Phi$, $[\alpha^{*}]\Phi$, $\langle \alpha^{*} \rangle \Phi$, $[\alpha^{*}]\Phi$

Formules régulières

- on peut autoriser des formules régulières, R
- Φ_{HMLReg} ::= tt | ¬Φ | Φ1 ∨ Φ2 | <R>Φ (plus les dérivés)
- R ::= ε| α | R1.R2 | R1+R2 | R^+ | R^*
- α ::= tt | ¬α | α1 ∨ α2 | a (plus les dérivés)
- interprétation :

<ε>Φ = Φ

<α>Φ et [α] Φ voir HML

<R1.R2>Φ = <R1><R2>Φ

[R1.R2]Φ = [R1][R2]Φ

<R1+R2>Φ = <R1>Φ ∨ <R2>Φ

[R1+R2]Φ = [R1]Φ ∧ [R2]Φ

$<R^*>\Phi = \Phi \vee <R>\Phi \vee <R><R>\Phi \vee \ldots$

$[R^*]\Phi = \Phi \wedge [R]\Phi \wedge [R][R]\Phi \wedge \ldots$

$<R^+>\Phi = <R.R^*>\Phi$

$[R^+]\Phi = [R.R^*]\Phi$

Formules régulières : exemples

- il est possible de faire la suite a,b,c : ?
- il est toujours possible de faire a : ?
- il sera un jour possible de faire a : ?
- Φ est toujours vrai : ?
- Φ sera vrai un jour : ?
- pas de blocage : ?
- il est impossible de faire deux ”entrée” sans faire un ”sortir” entre les deux :
  ?
- après l'envoi d'un message, il sera reçu un jour : ?

Formules régulières : exemples

- il est possible de faire la suite a,b,c : <a.b.c>tt
- il est toujours possible de faire a : [tt*]<a>tt
- il sera un jour possible de faire a : <tt*><a>tt
- Φ est toujours vrai : [tt*]Φ
- Φ sera vrai un jour : <tt*>Φ
- pas de blocage : [true*]<true>tt
- il est impossible de faire deux ”entrée” sans faire un ”sortir” entre les deux : [tt*.entrée.(¬sortir)*.entrée]ff
- après l'envoi d'un message, il sera reçu un jour : [envoi]<tt*.reception>tt

Le point et les objectifs

- on a vu comment décrire des systèmes (STE)
- on a vu comment décrire des propriétés
  - utilisation de logique(s) temporelles: (w)HML, rf-HML
  - propriétés de sûreté:
    - quelque chose de mal n'arrive jamais : [tt*] ¬Fmal
    - invariant : [tt*] Fbien
  - propriétés de vivacité:
    - quelque chose de bien finit toujours par arriver: <tt*> Fbien
  - combinaisons
- comment les vérifier en pratique ? → outils
- lps2bpes -f F.mcf S.lps S_F.bpes
  pbes2bool S_F.bpes

Sources d'éléments de transparents

- P. André et A. Vailly. *Conception des systèmes d'information, Panorama des méthodes et des techniques*, Ellipses, 2001.
  http://www.sciences.univ-nantes.fr/info/perso/permanents/andre/COURS/
- R. Mateescu. *Model Checking of Action-Based Concurrent Systems*, Summer school on Verification Technology, Systems and Applications (VTSA'08)
  http://www.mpi-inf.mpg.de/VTSA08/
- A. Mathijssen et al. *Behavioural Analysis using mCRL2*, IPA Course on Formal Methods, TU Eindhoven, 2008
  http://www.win.tue.nl/~fstapper/talks/ipacfm2008-06-26.pdf

Quelques outils

- CADP : http://www.inrialpes.fr/vasy/cadp/
- CWB-NC : http://www.cs.sunysb.edu/~cwb/
- LTSA : http://www.doc.ic.ac.uk/ltsa/
- MCRL2 : http://www.mcrl2.org/
- MEC4 : http://altarica.labri.u-bordeaux.fr/wiki/tools:mec_4
- SMV : http://www-2.cs.cmu.edu/~modelcheck/smv.html
- SPIN : http://spinroot.com/spin/whatispin.html

Bibliographie pour aller + loin

- A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*, Masson, 1992. [angl.,Prentice Hall, 1994]
- Ph. Schnoebelen, B. Bérard, M. Bidoit, F. Laroussinie et A. Petit. *Vérification de logiciels : techniques et outils du model-checking*, Vuibert, 1999. [angl.,Springer,2001]
- M. Huth et M. Ryan. *Logic in Computer Science. Modelling and reasoning about systems*, Cambridge Univ. Press, 2002.
- R. Mateescu. *Logiques temporelles basées sur actions pour la vérification des systèmes asynchrones*, TSI, 22(4):461-495, 2003. http://www.inrialpes.fr/vasy/Publications/Mateescu-03-b.html
- H. Garavel. *Défense et illustration des algèbres de processus*, 2003. http://www.inrialpes.fr/vasy/Publications/Garavel-03.html
- R. Milner. *Communication and Concurrency*, Prentice Hall, 1989.
- R. Cleaveland and S.A. Smolka. *Process Algebra,* Encyclopedia of Electrical Engineering. John Wiley & Sons, 1999.
- J. Magee et J. Kramer. Concurrency: State Models & Java Programs, 2nde édition, Wiley, 2006 http://www.doc.ic.ac.uk/~jnm/book/