Runtime monitoring

III. Spécification et vérification de contrats d'interface

Spécifications sur des séquences

Les logiques propositionnelle et du premier ordre nous permettent d'exprimer des contraintes sur un "instantané" de l'état d'un système...

...or on a vu qu'en runtime monitoring, un moniteur reçoit des séquences d'événements.

Comment spécifier des contraintes sur des séquences?

On verra deux langages d'entrée:

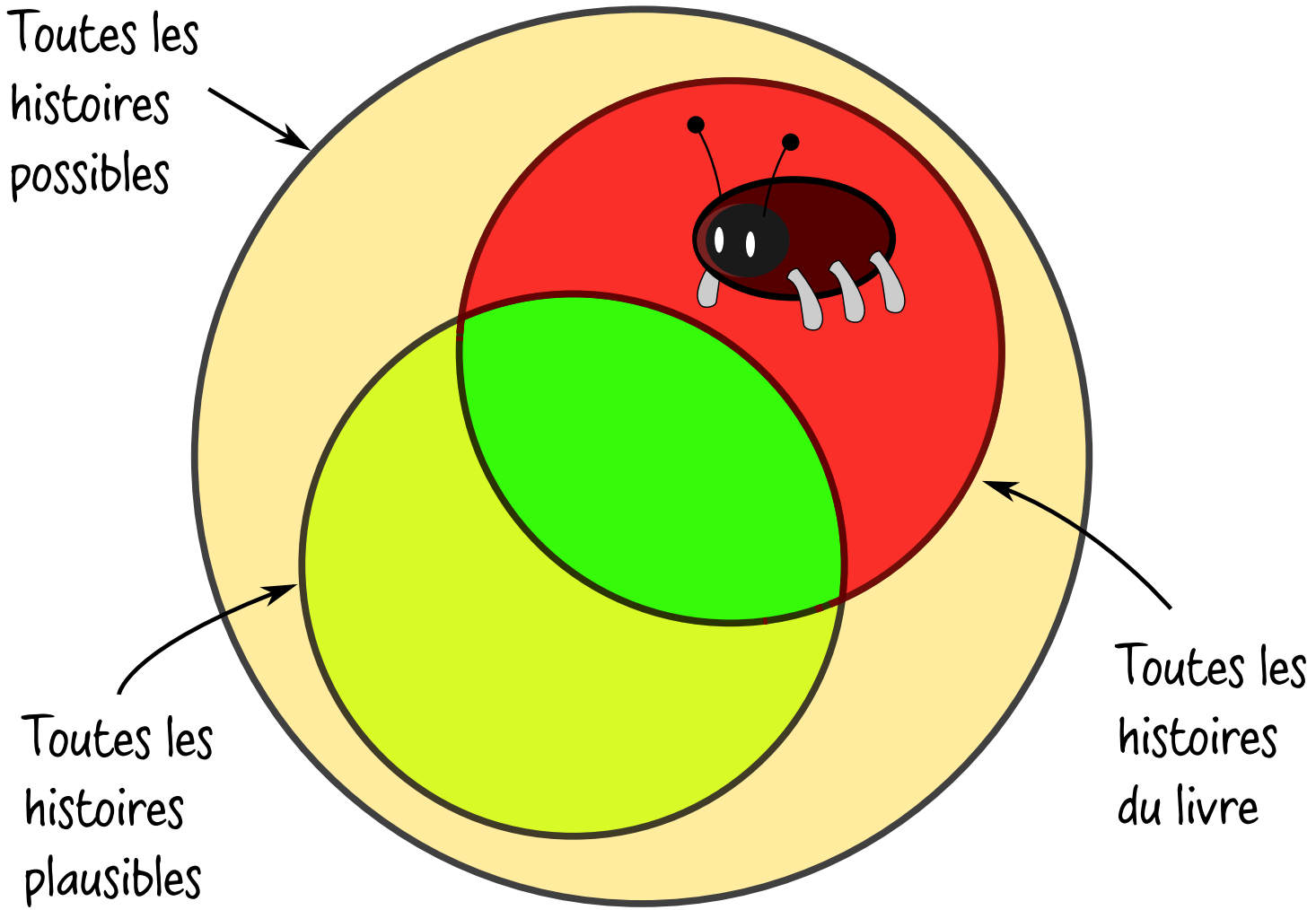
- Automates finis
- Logique temporelle linéaire (LTL)

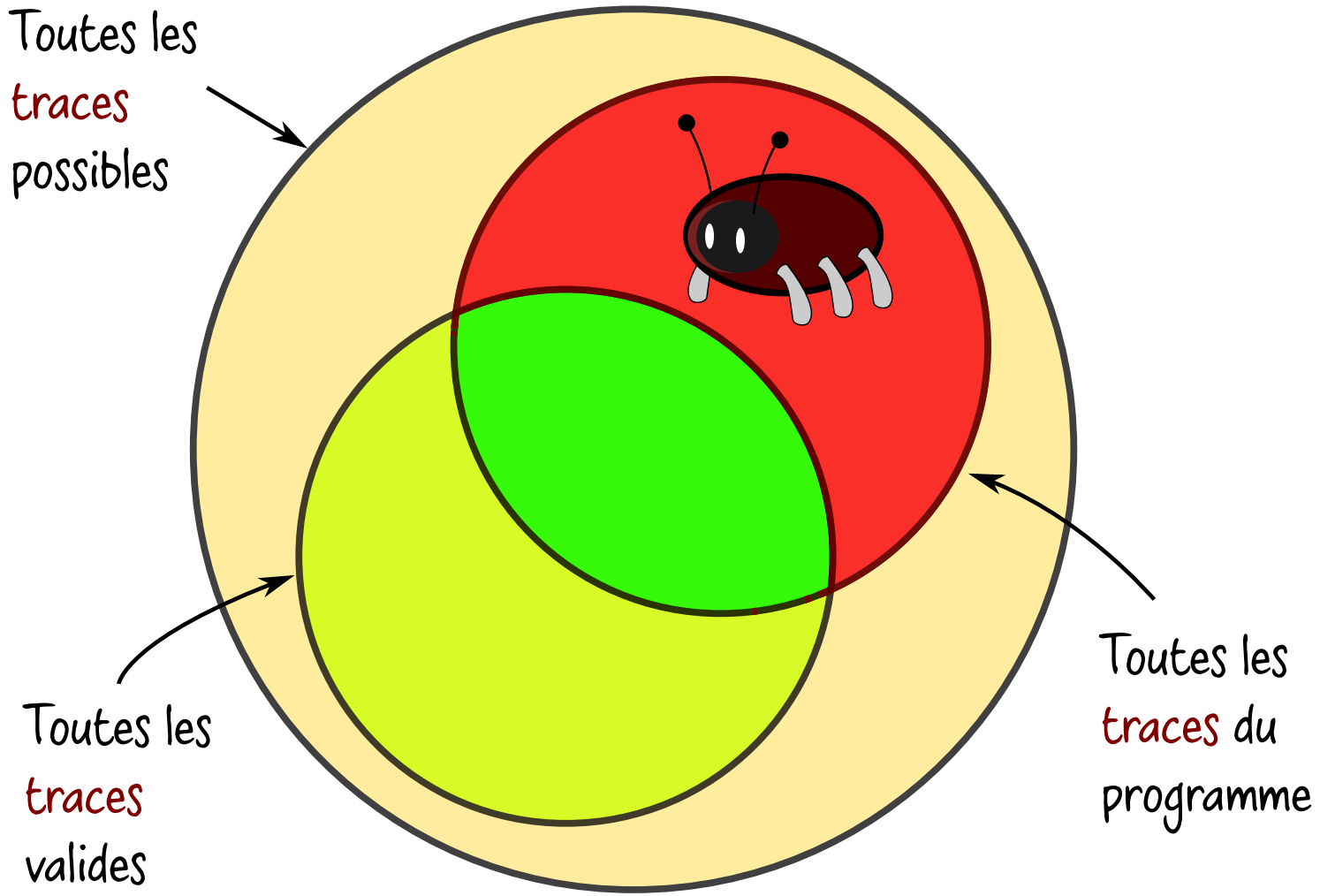
Associons à chaque événement distinct produit par un senseur un symbole. Alors:

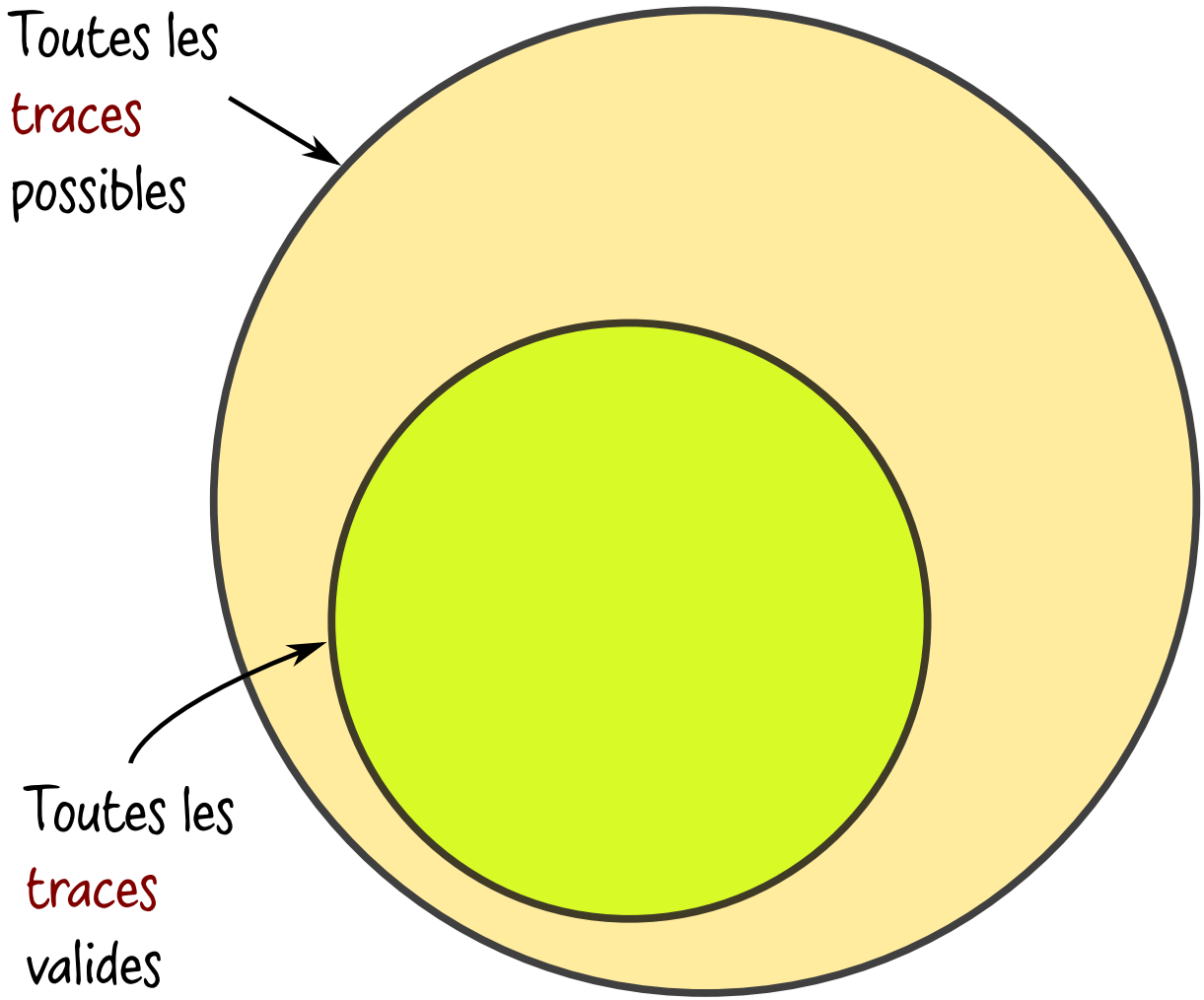
Alphabet (A)
Ensemble des symboles représentant chacun des événements possibles

Trace
Suite de symboles de A

Dans les exemples précédents, quel est l'alphabet?
Pouvez-vous donner un exemple de trace?


Toutes les histoires possibles
Toutes les histoires plausibles
Toutes les histoires du livre


Toutes les
traces
possibles
Toutes les
traces
valides
Toutes les
traces du
programme


Toutes les
traces
possibles
Toutes les
traces
valides

Une spécification φ propre au monitoring doit décrire, parmi toutes les traces possibles, lesquelles sont souhaitées ou attendues

Si on a...

$X \models \varphi$

Spécification de traces

Qu'est-ce que X? Autrement dit, que seront les modèles de φ?

En général, on désigne l'alphabet par Σ ou A. Alors:

- Σ* (ou A*) désigne l'ensemble de toutes les traces fabriquées à partir de Σ (resp. A)

C'est un ensemble infini!

- Une trace est donc un élément de Σ* ; on désigne souvent une trace par le symbole σ
- Convention de notation: σ_i est le i-ème symbole de la trace σ, et σ^i est le suffixe de σ en commençant au i-ième symbole
- Une spécification de traces représente donc un (sous-)ensemble $S \subseteq \Sigma^*$

Comment spécifier les traces attendues?

Méthode #1: on les énumère (toutes!)

Symbole désignant la trace vide

S = {ε,
hasMoreElements,
hasMoreElements nextElement,
hasMoreElements hasMoreElements,
hasMoreElements nextElement
hasMoreElements,
...}

Fastidieux!

Méthode #2: on utilise un automate fini étiqueté

$$M = <S, s_0, A, F, \delta>$$

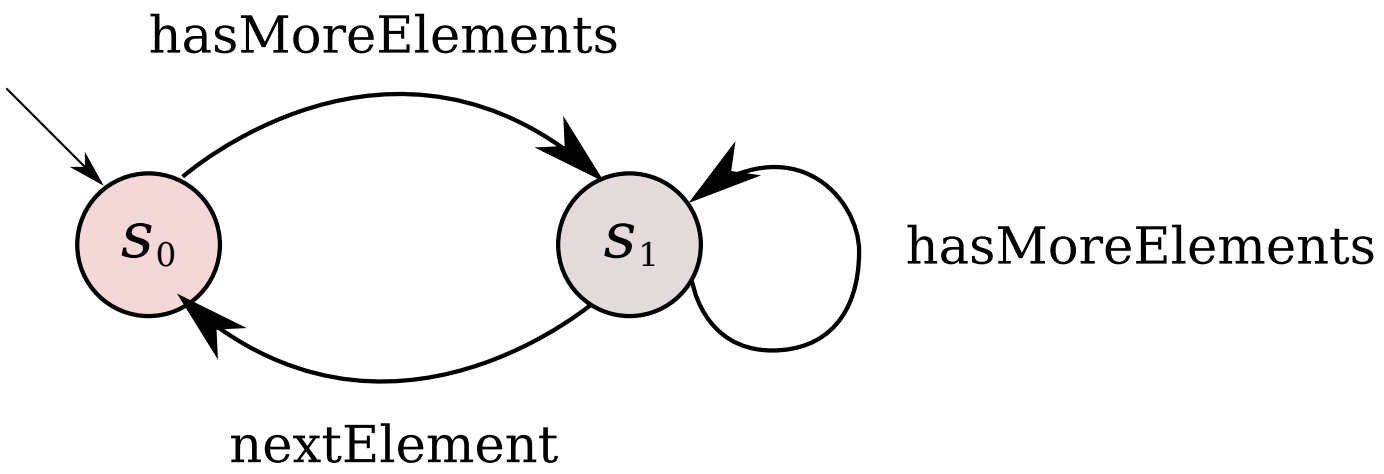
Ensemble d'états

État initial

Étiquettes des arcs

États finaux

Relation de transition

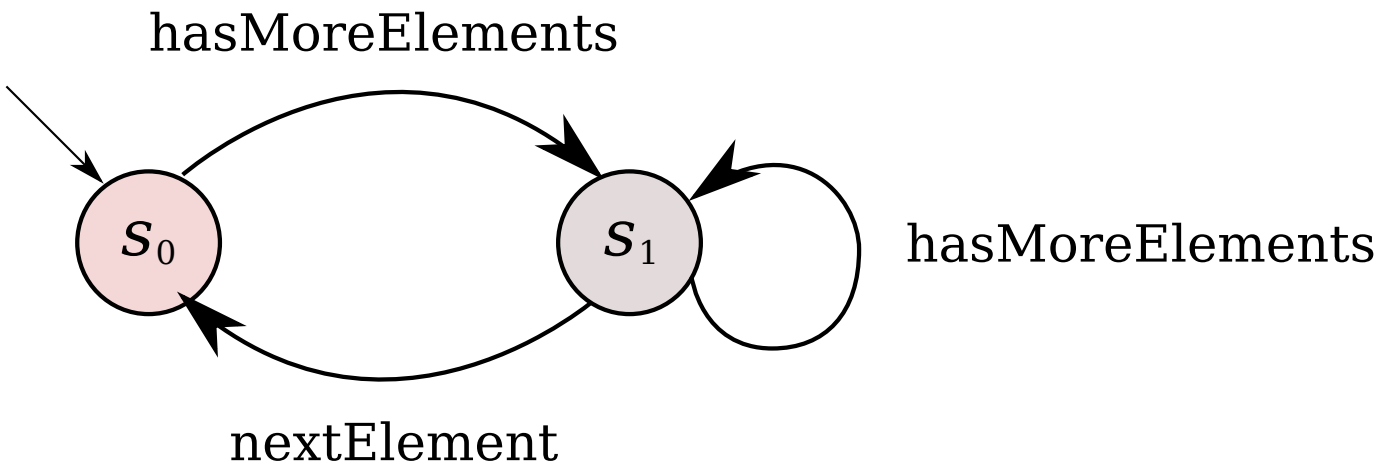


La relation de transition est définie comme:

$$\delta : S \times A \rightarrow S$$

Dans l'exemple ci-dessous...

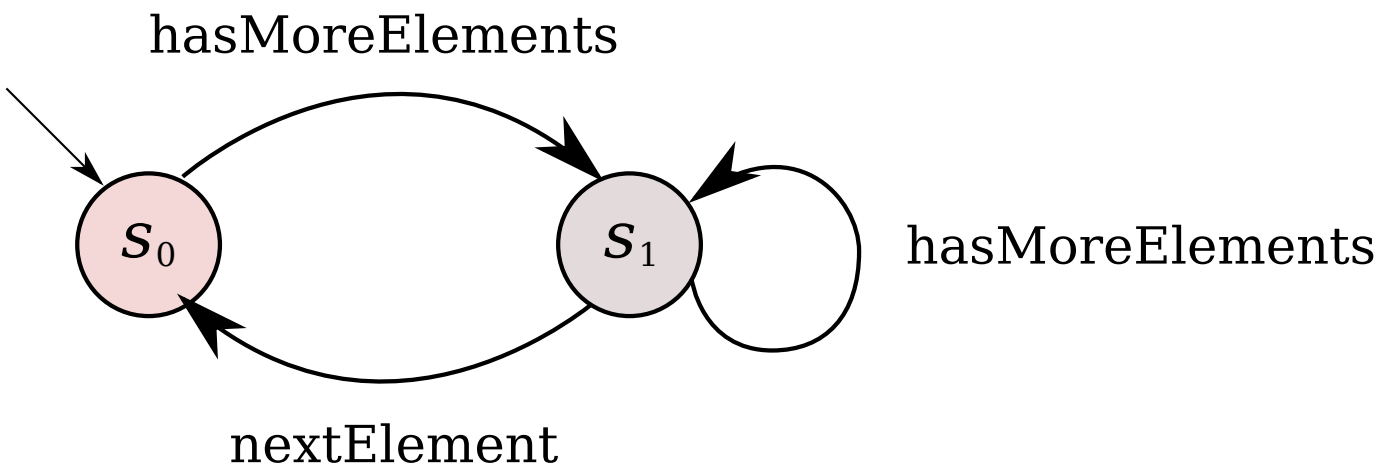
- Que retourne $\delta(s_0$, hasMoreElements)?
- Que retourne $\delta(s_1$, nextElement)?



Le langage défini par un automate M, noté $L(M)$, est l'ensemble des traces $\sigma \in \Sigma^*$ telles que $\delta(s_0, \sigma) \in F$.

Autrement dit, en "lisant" les symboles de σ un à un, suivre δ nous fait aboutir à un état final de M.

Quels devraient être les états finaux dans l'automate ci-dessous?



Automate vu comme une expression logique

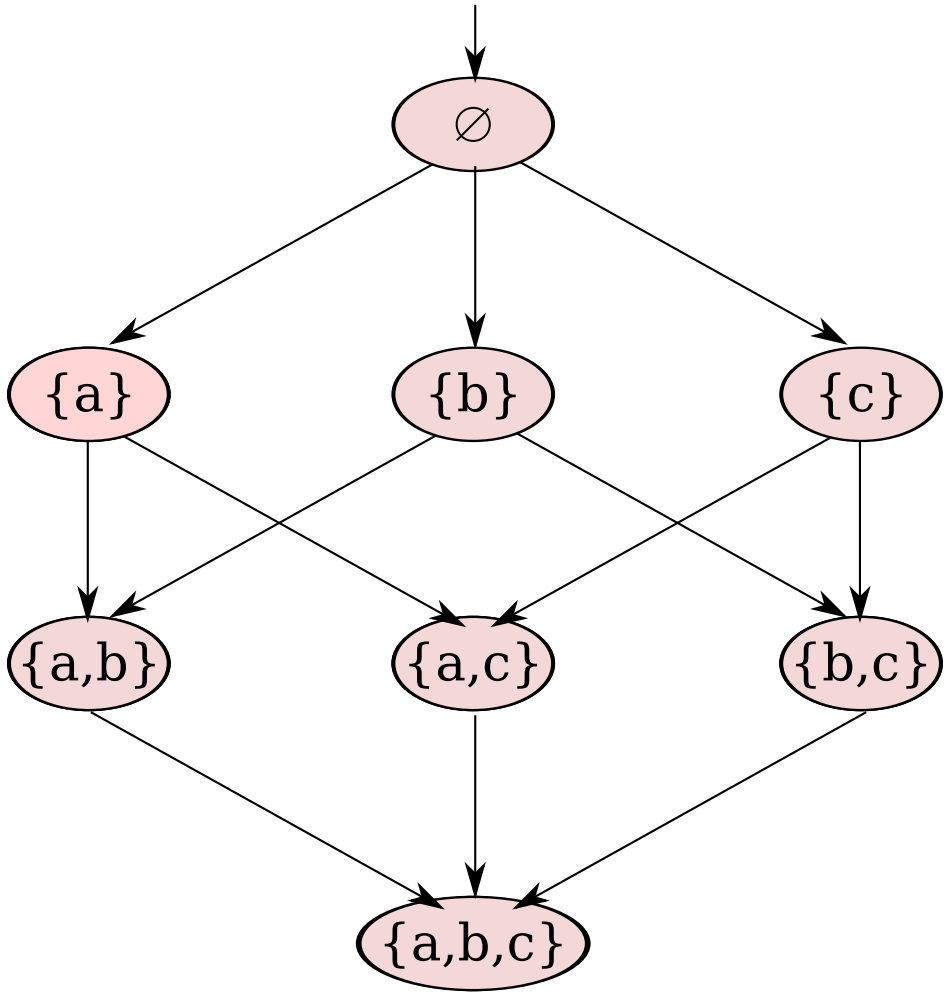
"Abusons" de la notation et écrivons $\sigma \models M$ lorsque $\delta(s_0, \sigma) \in F$.

Qu'arrive-t-il aux problèmes déjà connus...

- Pour une trace σ donnée, comment évalue-t-on que $\sigma \models M$?
- Que signifie "M est satisfaisable"?
- Comment le détermine-t-on?

Comment spécifie-t-on les contrats d'interface vus précédemment?

Considérons l'alphabet $\Sigma = \{a,b,c\}$ et le contrat d'interface: "a, b et c doivent éventuellement survenir"



Extension de la logique propositionnelle dont les modèles sont des traces sur un alphabet Σ

- Les énoncés élémentaires sont des symboles de Σ
- Tous les connecteurs de la logique propositionnelle sont présents
- On y ajoute quatre opérateurs temporels: **F**, **G**, **X**, **U**
- Expression LTL = affirmation sur la succession des symboles

Amir Pnueli
(1941-2009)

Signification intuitive des opérateurs:

F φ	"éventuellement φ" (*Future*)
G φ	"toujours φ" (*Globally*)
X φ	"φ pour le prochain symbole" (*neXt*)
φ **U** ψ	"φ jusqu'à ce que ψ" (*Until*)

Les expressions suivantes sont-elles vraies ou fausses pour les traces données?

- abacadacbac
- bcdbcbdbcbd

a) **F** a b) **G** a c) ¬d **U** c

Signification intuitive des opérateurs:

F φ	"éventuellement φ" (*Future*)
G φ	"toujours φ" (*Globally*)
X φ	"φ pour le prochain symbole" (*neXt*)
φ **U** ψ	"φ jusqu'à ce que ψ" (*Until*)

Les expressions suivantes sont-elles vraies ou fausses pour les traces données?

- abacadacbac
- bdcbcbdbcbd

a) **F** a b) **G** a c) ¬d **U** c

Les opérateurs temporels peuvent être imbriqués et composés avec les connecteurs de logique propositionnelle

Les expressions suivantes sont-elles vraies ou fausses pour les traces données?

- abacadacbac
- bdcbcddbcbd

a) $\mathbf{G}\ (b \vee c \vee d)$ b) $\mathbf{G}\ (b \rightarrow (\mathbf{X}\ a))$ c) $\mathbf{F}\ \mathbf{G}\ a$

Tout comme avec les autres logiques, on peut donner des règles récursives qui permettent d'évaluer une expression LTL sur une trace σ donnée:

$\sigma \models a$	lorsque	$\sigma_1 = a$
$\sigma \models \neg\varphi$	lorsque	$\sigma \not\models \varphi$
$\sigma \models \varphi \wedge \psi$	lorsque	$\sigma \models \varphi$ et que $\sigma \models \psi$
$\sigma \models \varphi \vee \psi$	lorsque	$\sigma \models \varphi$ ou $\sigma \models \psi$
$\sigma \models \varphi \rightarrow \psi$	lorsque	$\sigma \not\models \varphi$ ou $\sigma \models \psi$
$\sigma \models \mathbf{G}\,\varphi$	lorsque	$\sigma_1 \models \varphi$ et $\sigma^2 \models \mathbf{G}\,\varphi$
$\sigma \models \mathbf{F}\,\varphi$	lorsque	$\sigma_1 \models \varphi$ ou $\sigma^2 \models \mathbf{F}\,\varphi$
$\sigma \models \mathbf{X}\,\varphi$	lorsque	$\sigma_2 \models \varphi$
$\sigma \models \varphi\ \mathbf{U}\ \psi$	lorsque	$\sigma_1 \models \psi$ ou alors $\sigma_1 \models \varphi$ et $\sigma^2 \models \varphi\ \mathbf{U}\ \psi$

Comment évalue-t-on:

$$\sigma \models \mathbf{F}\ a$$

lorsque $\sigma = b$?

Si on suit la définition, $\sigma \models \mathbf{F}\ a$ lorsque soit:

1) $\sigma_1 \models a$ ← Faux!

ou

2) $\sigma^2 \models \mathbf{F}\ a$

La réponse dépend donc du contenu de σ^2 ... mais si on doit retourner une réponse maintenant?

Après le premier symbole, **F** a n'est "ni vraie ni fausse"...
il nous faut donc une troisième valeur... "indéterminé"

Nécessaire seulement pour évaluer un préfixe fini