DIAGRAMME ÉTATS-TRANSITIONS

(*Statecharts ou State machines*)

Diagramme comportemental

Modélisation du cycle de vie des éléments fortement réactifs

SOMMAIRE

- Introduction
- Pour quelles classes ?
- Comment le construire ?
- Comment le représenter ?
- État
- Transition
- Événement
- Message
- Condition
- Action
- Activité

INTRODUCTION

- Les diagrammes d'états sont **utilisés** pour compléter aussi bien :
  - le diagramme de classes d'analyse que
  - le diagramme de classes technique.

- Ils sont élaborés **pour une classe** afin de visualiser le comportement d'un objet au cours du temps.

INTRODUCTION

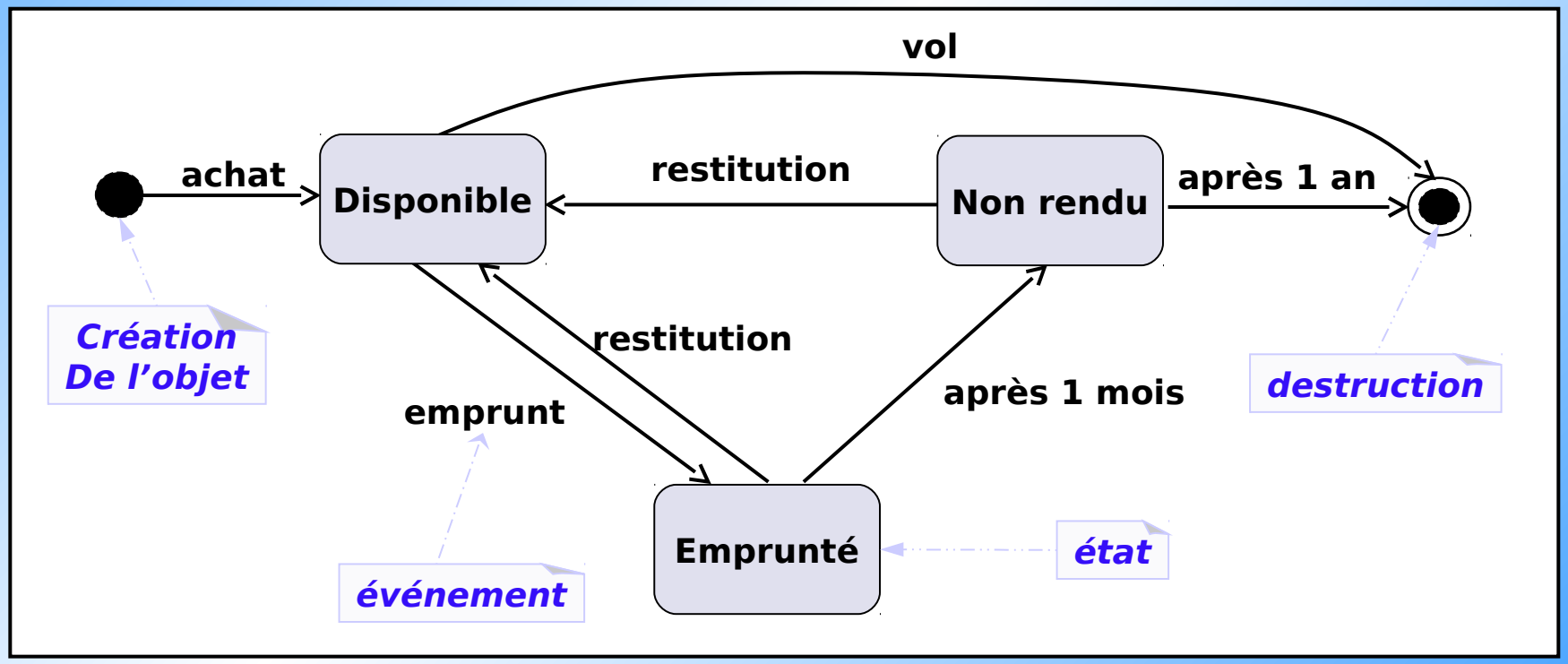
- UML a repris le concept de *machine à états finis* : qui consiste à s'intéresser au **cycle de vie d'une instance générique d'une classe** particulière au fil de ses interactions avec le reste du monde, dans tous les cas possibles.

- Cette vue locale d'un objet, qui **décrit comment il réagit à des événements** en fonction de son état courant et comment il passe dans un nouvel état, est représentée graphiquement sous la forme d'un *diagramme d'états*.

INTRODUCTION

Exemple de diagramme d'états

Bibliothèque – objet exemplaire



INTRODUCTION

- Le diagramme d'états représente le cycle de vie des objets d'une classe.

- Les objets subissant les contraintes extérieures passent par une succession d'états **modifiant la valeur des attributs** ou des **règles composant l'objet**.

ÉTATS D'UN OBJET

Ils sont fonction de :

- **La valeur des propriétés de l'objet**

  ex : un élève est dit *"recalé "* si sa moyenne est inférieure à *10/20*.

- **La valeur des propriétés des objets qui lui sont reliés**

  ex : un client est dit *"suspendu "* si le solde d'un de ses comptes est négatif depuis plus de *5* jours.

- **La valeur des états des objets qui lui sont reliés**

  ex : un client est dit *"inactif "* s'il n'existe pas pour ce client d'occurrences de relation avec un contrat à l'état *"en cours "*.

ÉTATS D'UN OBJET

Ils sont fonction de :

- **Ses relations ou de son absence de relation avec d'autres objets**

  ex : une commande est dite *"en attente "* s'il n'existe pas pour cette commande d'occurrences de relation avec une livraison.

- **La valeur des propriétés de ses relations avec d'autres objets**

  ex : un produit est dit *"en rupture de stock "* dans un dépôt si la quantité en stock < seuil critique.

INTRODUCTION

- Les diagrammes d'états constituent une **modélisation du comportement d'un objet**.

- Un diagramme d'états relie des événements à des états en spécifiant la **séquence d'états** provoquée par une **séquence d'événements** (*ou déclencheurs ou triggers*).

- Une transition représente une modification d'état provoquée par un événement.

POUR QUELLES CLASSES ?

Toutes les classes du modèle statique ne requièrent pas nécessairement une machine à états, représentée par un diagramme d'états.

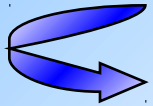
Un diagramme d'états n'a d'utilité que si l'**objet passe par au moins trois états**.

POUR QUELLES CLASSES ?

- Classes qui ont un **comportement dynamique complexe** nécessitant une description poussée.

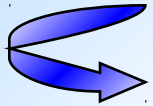
Cela correspond à l'un des deux cas suivants :

1- **Les objets de la classe peuvent-ils réagir différemment à l'occurrence du même événement** ?



Chaque type de réaction caractérise un état particulier.

2- **La classe doit-elle organiser certaines opérations dans un ordre précis** ?



états séquentiels précisant la chronologie forcée des événements d'activation.

COMMENT LE CONSTRUIRE ?

1- Représenter tout d'abord la **séquence d'états** qui décrit le **comportement nominal d'un objet**, avec les transitions qui y sont associées.

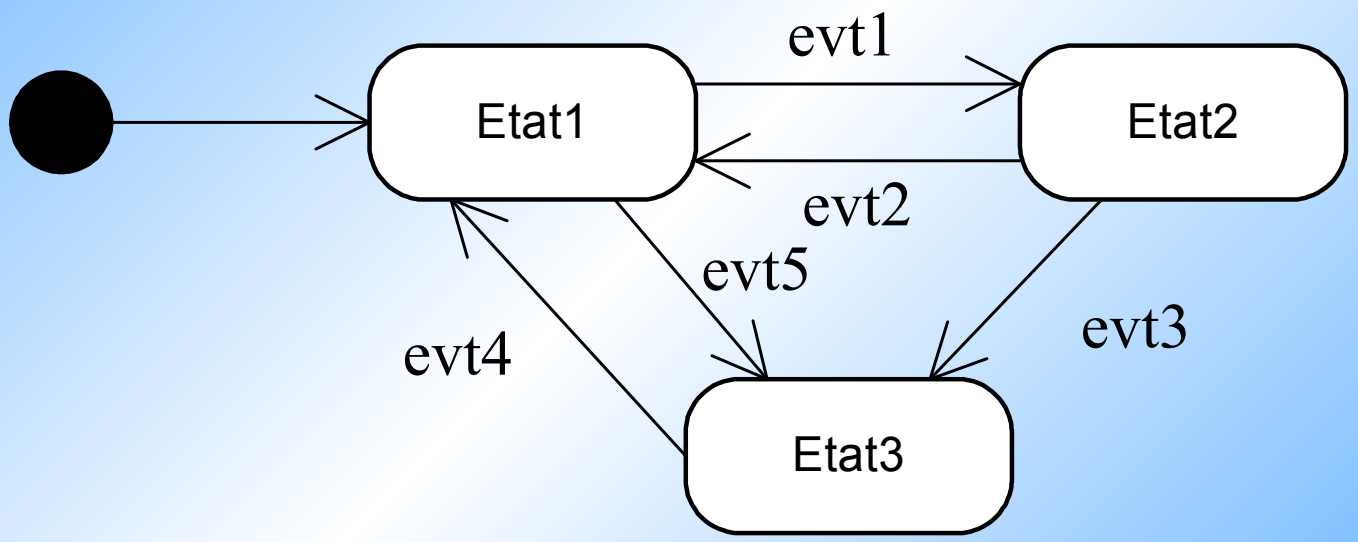
2- Ajouter progressivement les transitions qui correspondent aux **comportements alternatifs ou d'erreur**.

3- Compléter les **actions** sur les transitions et les **activités** dans les états.

4- **Structurer le diagramme** en sous-états s'il devient trop complexe.

COMMENT LE REPRÉSENTER ?

- Un diagramme d'états est un **graphe** dont les **nœuds** sont des **états** et les **arcs** orientés des **transitions** désignées par les noms d'événements.



- Le passage d'un état à un autre est **instantané** car le système doit toujours être dans un état connu.

ÉTAT

- Un *état* représente une situation durant la vie d'un objet pendant laquelle :
  - ✓ il satisfait une certaine **condition**,
  - ✓ il exécute une certaine **activité**,
  - ✓ ou bien il attend un certain **événement**.

- Un objet passe par une succession d'états durant son existence. Un état a une **durée finie**, **variable** selon la vie de l'**objet**, en particulier en fonction des événements qui lui arrivent.

COMMENT IDENTIFIER LES ÉTATS ?

Comment trouver les états d'une classe ?

Pour faire un parallèle, on peut dire qu'il est **aussi difficile** de trouver les bons états dans le modèle dynamique que les bonnes **classes** dans le modèle statique !

COMMENT IDENTIFIER LES ÉTATS ?

Il n'y a donc pas de recette miracle, cependant **trois démarches complémentaires** peuvent être mises en œuvre :

1- La **recherche intuitive** repose sur l'expertise métier. Certains états fondamentaux font partie du vocabulaire des experts du domaine et sont identifiables a priori.

2- L'étude des attributs et des associations de la classe peut donner des indications précieuses : chercher des **valeurs seuils d'attributs** qui modifient la dynamique, ou des **comportements** qui sont induits par **l'existence** ou **l'absence** de **certains liens**.

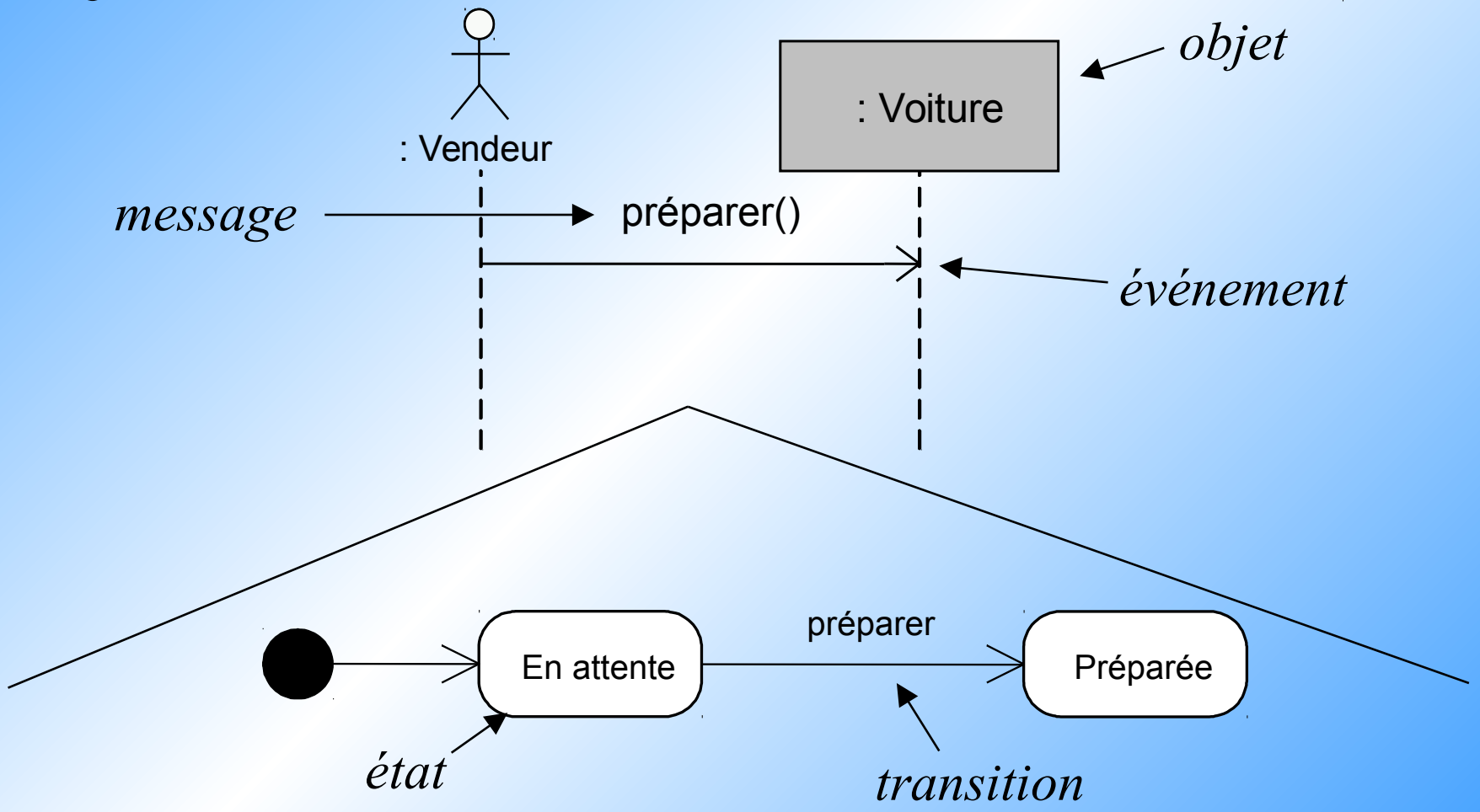
COMMENT IDENTIFIER LES ÉTATS ?

3- Une **démarche systématique** peut également être utilisée, classe par classe :

- ✓ chercher le diagramme d'interaction le plus représentatif du comportement des instances de cette classe,
- ✓ associer un état à chaque intervalle entre événements émis ou reçus par une instance et
- ✓ placer les transitions.
- ✓ reproduire ensuite cette démarche avec tous les scénarios faisant intervenir des instances de la classe, afin d'ajouter de nouvelles transitions ou de nouveaux états.

La difficulté principale consiste à trouver ensuite les boucles dans le diagramme, afin de ne pas multiplier les états.

Comment représenter les concepts dynamiques de base ?



ÉTAT INITIAL ET ÉTAT FINAL

- ***2* pseudo-états** :

  ✓ l'**état initial** du diagramme d'états : **création** de l'instance.

  ✓ l'**état final** du diagramme d'états : **destruction** de l'instance.

Événement / transition

- Un **événement** spécifie qu'il s'est passé quelque chose de significatif, localisé dans le temps et dans l'espace.

- Un **événement** représente l'occurrence d'un stimulus qui peut déclencher une transition entre états.

- Une **transition** décrit la réaction d'un objet lorsqu'un événement se produit (*généralement l'objet change d'état*).

ÉVÉNEMENT

- *6* sortes d'événements (*ou déclencheurs ou triggers*) :

1- La **réception d'un message** (*signal event*) envoyé par un autre objet, ou par un acteur. L'envoi d'un message est en général asynchrone (*l'émetteur n'est pas bloqué et peut continuer son exécution*).

2- L'**appel d'une opération** (*call event*) sur l'objet récepteur. L'événement d'appel est en général synchrone (*l'émetteur est bloqué et attend que l'appelé ait fini de traiter le message)*.

Ex : calculerMontant() (*message synchrone*)
imprimer() (*message asynchrone*)

ÉVÉNEMENT

3- Le **passage du temps** (*time event*), qui se modélise en utilisant le mot-clé *after* suivi d'une expression représentant une durée, décomptée à partir de l'entrée dans l'état courant.
Ex : after(durée), after(3 minutes)

4- Un **changement** dans la satisfaction d'une condition (*change event*). On utilise alors le mot-clé *when*, suivi d'une expression booléenne. L'événement de changement se produit lorsque la condition passe à vrai.
Ex : when (solde <= 1 000 €)
Différence avec une condition de garde : un événement de changement est évalué continuellement jusqu'à ce qu'il devienne vrai, et c'est à ce moment-là que la transition se déclenche.

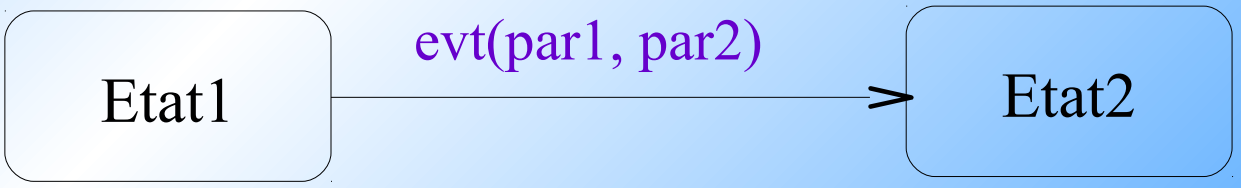
ÉVÉNEMENT

5- La **fin d'une activité** (*completion event*) de type *do/*, interne à un état. Cela peut déclencher une transition dite automatique qui ne porte pas de déclencheur explicite.

6- **at (date)**

Ex : at (date = 2 / 10 / 2016)

Un **événement** peut porter des **paramètres** qui matérialisent le flot d'informations ou de données entre objets.

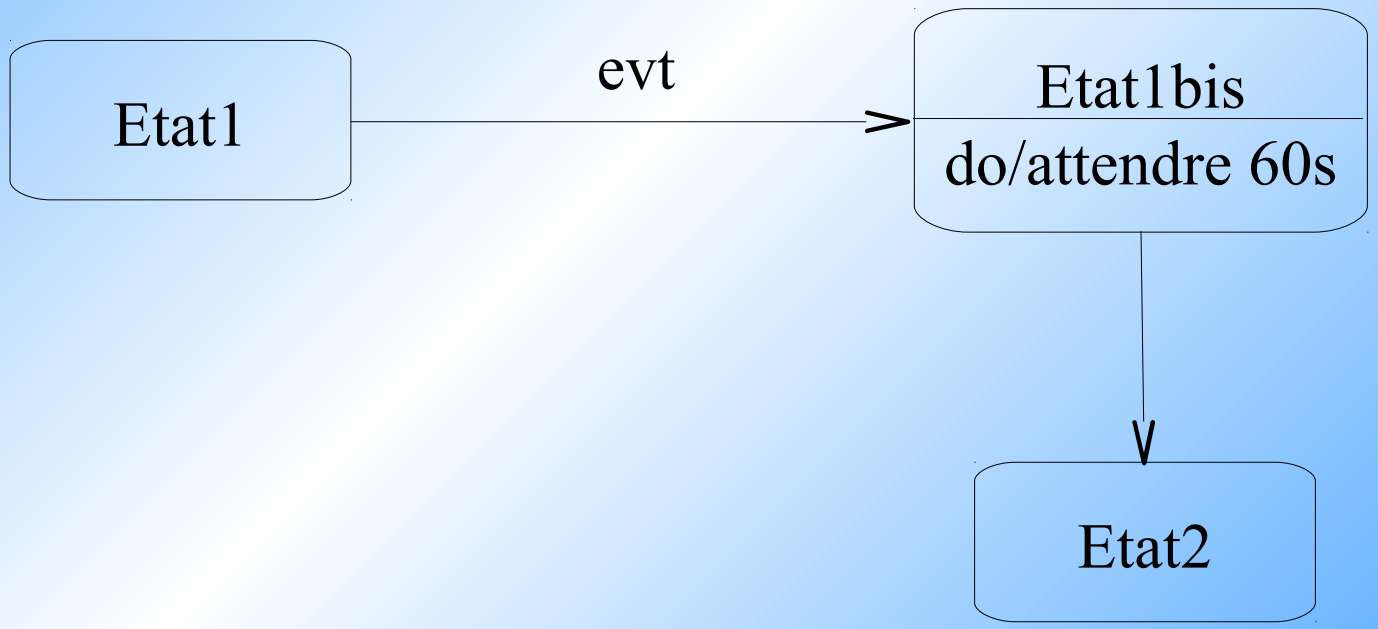


EXERCICE

Que pensez-vous de ce modèle ?

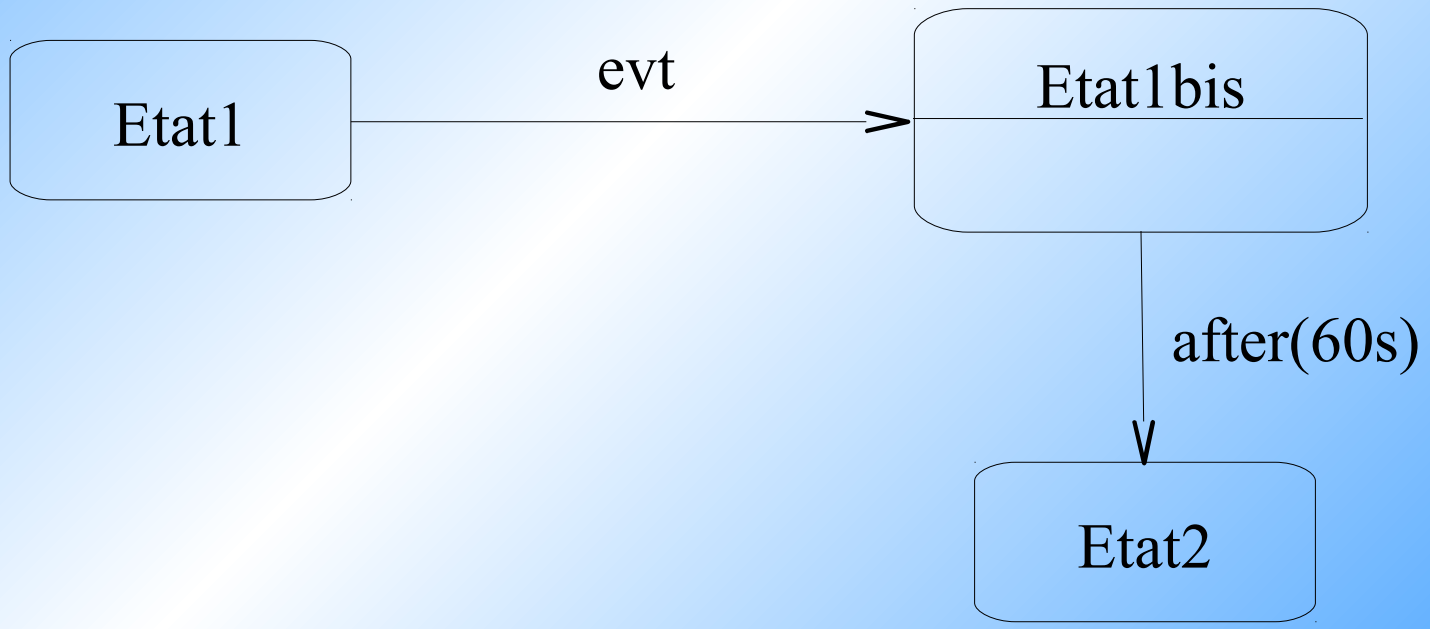
SOLUTION

Transition doit être instantanée



SOLUTION

Transition doit être instantanée



MESSAGE

- Un **message** est une transmission d'information unidirectionnelle entre deux objets, l'objet émetteur et l'objet récepteur.

- Le **mode de communication** peut être :
  - **synchrone** ou
  - **asynchrone**.

- La réception d'un message est un événement qui doit être traité par le récepteur.

CONDITION

Une **condition** est une expression booléenne qui doit être vraie lorsque l'événement arrive pour que la transition soit déclenchée.

OU

Les gardes

- Une **condition de garde** est une expression booléenne.
- Une **condition de garde** doit être vraie pour qu'une transition gardée ait lieu quand survient un événement.
- Une **condition gardée** est symbolisée par une expression booléenne entre crochets :

événement(attributs)[condition]

Exemples

- infraction [nombre points < 20]
- désistement [assez de places]

OPÉRATION

- La description comportementale d'un objet doit spécifier ce que fait l'objet en réponse aux événements.

- Des opérations attachées aux états ou aux événements sont exécutées en réponse aux états ou aux événements correspondants.

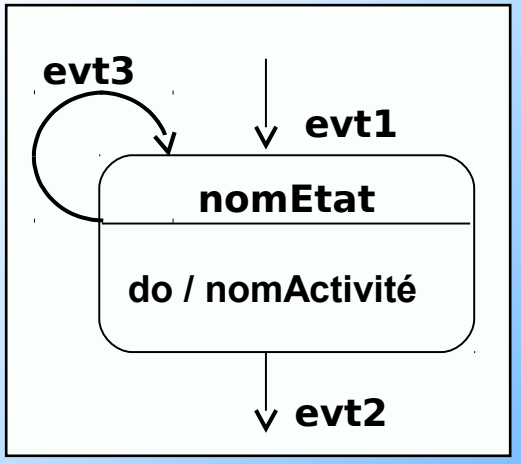
- *2* types d'**opérations** :
  - *les **activités***
  - *les **actions***

ACTIVITÉ

- Une **activité** est une opération qui nécessite du temps pour s'exécuter.

- Une **activité** est associée à un état.

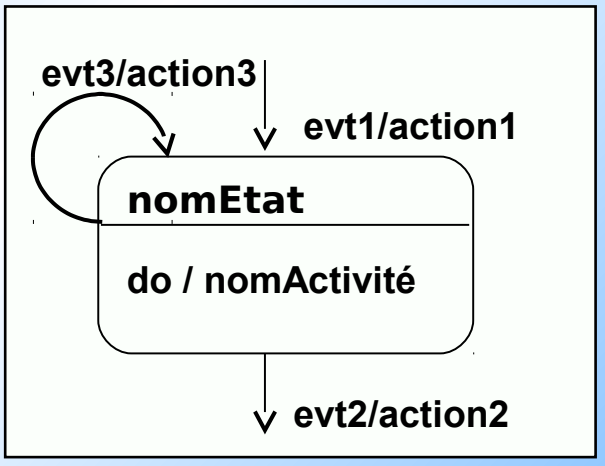
- Une **activité** est exécutée dans un état tant que l'on se trouve dans cet état, ou jusqu'à ce que le calcul associé soit terminé.



- Une **activité** est placée à l'intérieur d'une boite d'état après un *do* : *"do / nomActivité "*

ACTION

- Une **action** est une opération instantanée.
- Une **action** est associée à un événement.



- Une **action** est généralement exécutée pendant une transition.
- Une **action** est placée après l'événement séparée par une barre oblique : "*nomEvénement / nomAction*".

Action / Activité

- Les **actions** sont associées aux transitions et sont considérées comme **atomiques**, c'est-à-dire ininterruptibles, ou encore instantanées par rapport à l'échelle de temps considérée.

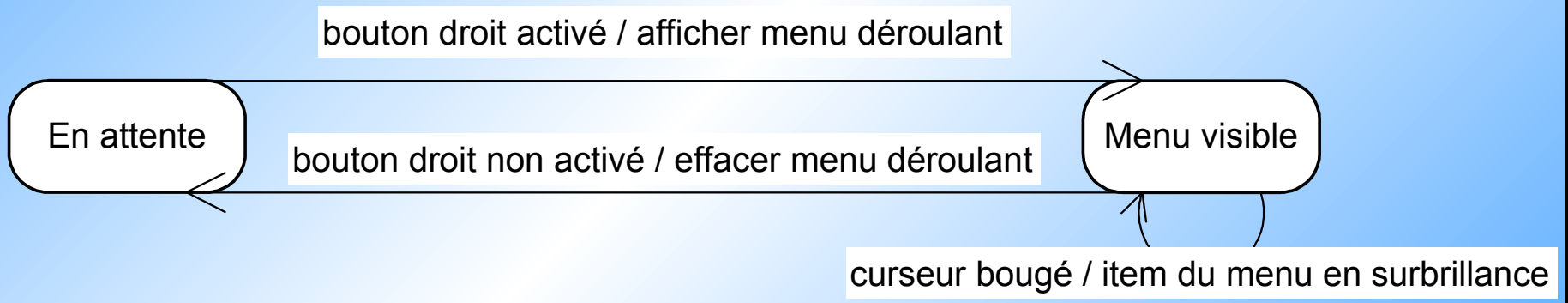
- Elles peuvent représenter des mises à jour de valeurs d'attributs, appels d'opérations, la création ou la destruction d'un autre objet, ainsi que l'envoi d'un signal à un autre objet.

- Les **activités**, au contraire des actions, ont une certaine durée, sont **interruptibles** et sont associées aux états.

- Les notions de durée et d'atomicité étant contextuelles et relatives, le travail du modélisateur consiste à choisir le bon concept au moment opportun.

EXEMPLE

Diagramme d'états d'un menu déroulant sur une station de travail

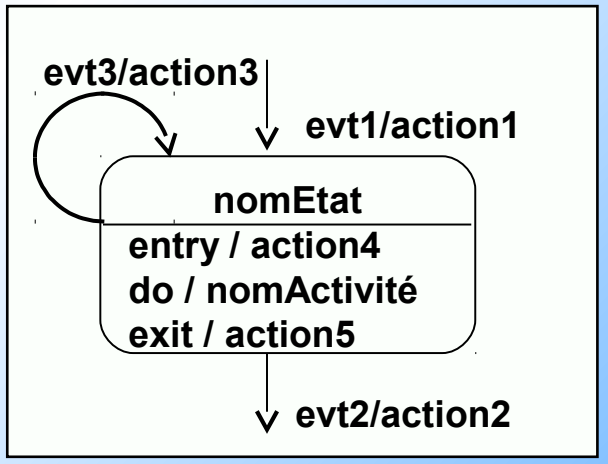


Action d'entrée / sortie

- Une **action en entrée** (*sortie*) est exécutée chaque fois que l'on rentre (*quitte*) dans l'état.

- Elles sont utiles lorsque toutes les transitions qui entrent (*sortent*) dans un état utilisent une **action commune**.

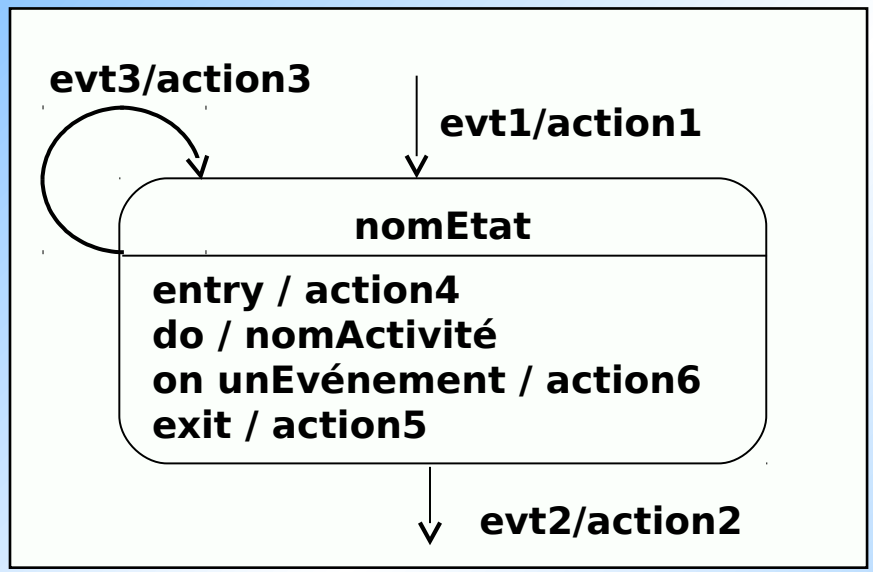
- L'action est indiquée après les mots clés "*entry* " ou "*exit* " suivis du caractère "/ ".



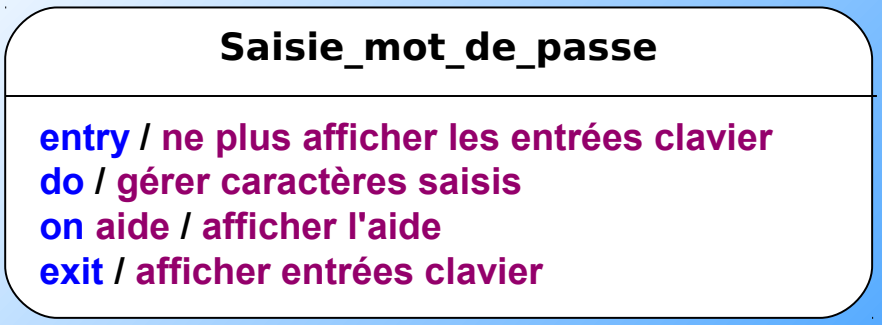
ACTION

- Un état peut contenir une autre **action** exécutée lors de l'occurrence d'un événement pendant que l'**objet est dans l'état** : c'est un **événement interne**.

- L'action est indiquée après le mot clé *"on unEvénement / "*.

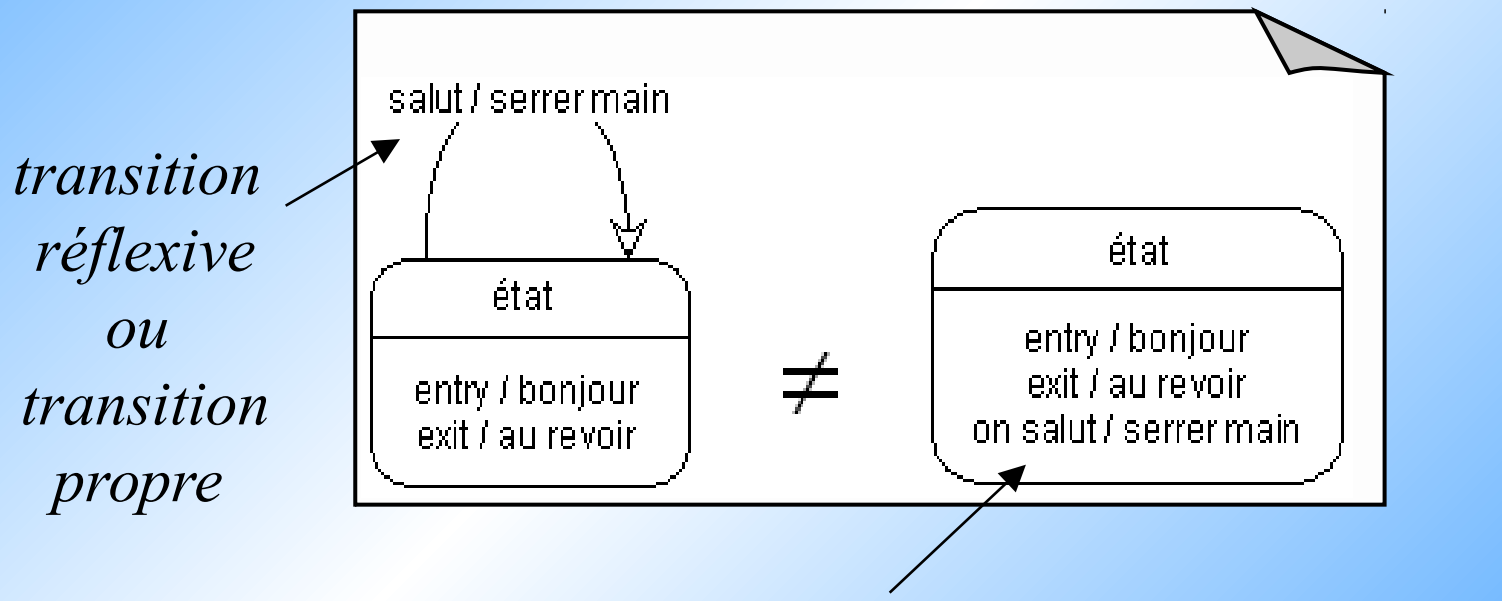


Exemple

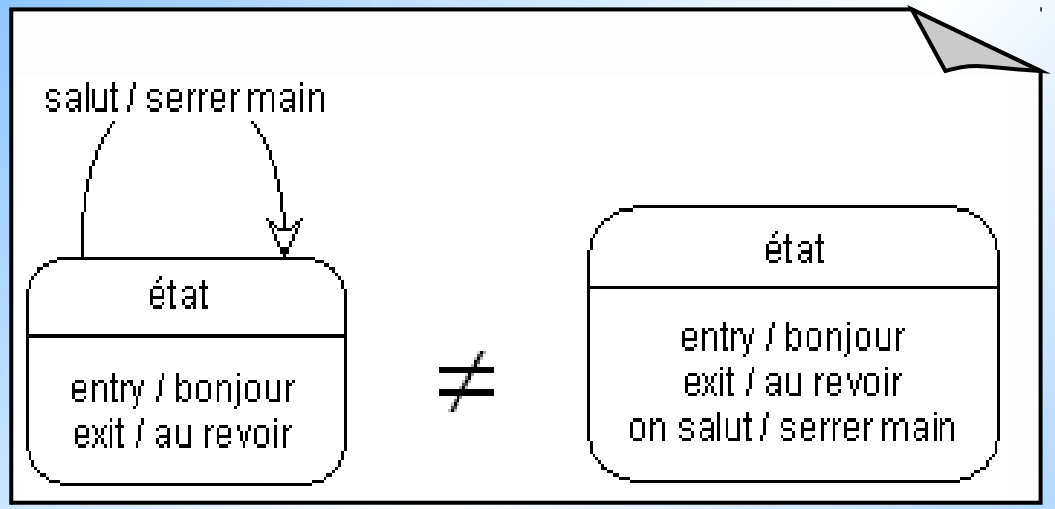


ÉVÉNEMENT INTERNE

- Un **événement interne** n'entraîne pas l'exécution des actions d'entrée et de sortie, contrairement au déclenchement d'une transition réflexive.

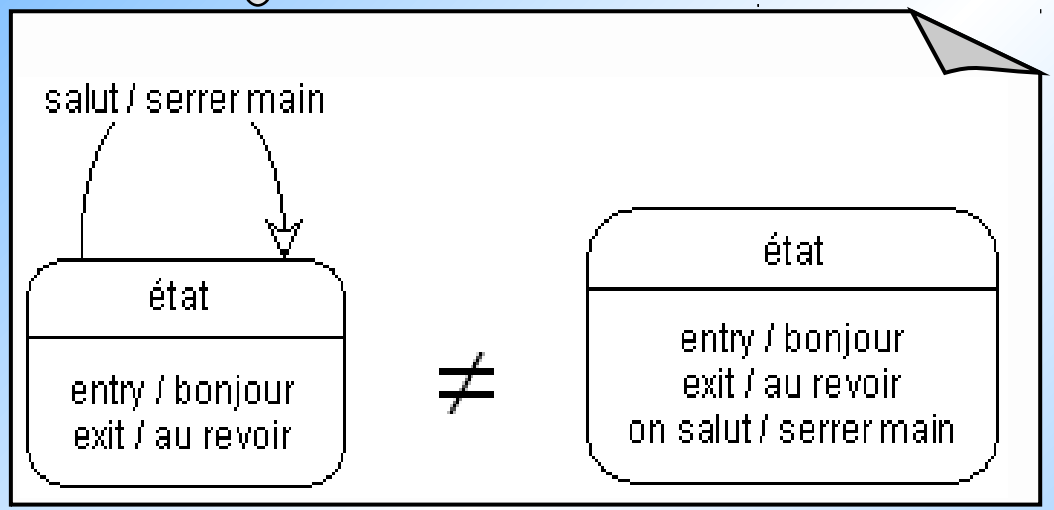


EXERCICE



On rentre dans l'état "*état* " et l'événement "*salut* " arrive.
Que se passe-t-il pour les deux modèles ?

SOLUTION



- bonjour
- evt salut → au revoir
  serrer main
  bonjour
- evt salut → au revoir
  serrer main
  bonjour

- bonjour
- evt salut → serrer main
  …
- evt salut → serrer main

EXÉCUTION OPÉRATION

- Si plusieurs opérations sont spécifiées dans un état, elles sont **exécutées dans l'ordre suivant** :

évt1 survient

- ✓ action sur la transition d'entrée

- ✓ action d'entrée dans l'état
- ✓ activité *"do"* dans l'état

entrée dans l'état

- ✓ action associée aux événements internes

- ✓ action de sortie de l'état
- ✓ action sur la transition de sortie de l'état

évt2 survient

A savoir

DIAGRAMME D'ÉTATS

- Un diagramme d'états peut **s'exécuter une seule fois** ou en **boucles continues**.

- Les diagrammes qui **s'exécutent une seule fois** représentent les objets qui ont une **vie finie**. Ils possèdent un **état initial** et un **état final**.

LECTURE DES DIAGRAMMES D'ÉTATS

- Si un **objet** est **dans un état** et qu'un **événement** indiqué sur l'une des transitions **se produit**, alors l'**objet** entre dans un **nouvel état**.

- Si **plus d'une transition** sortent d'un état, alors le premier événement survenant provoque le franchissement de la transition correspondante.

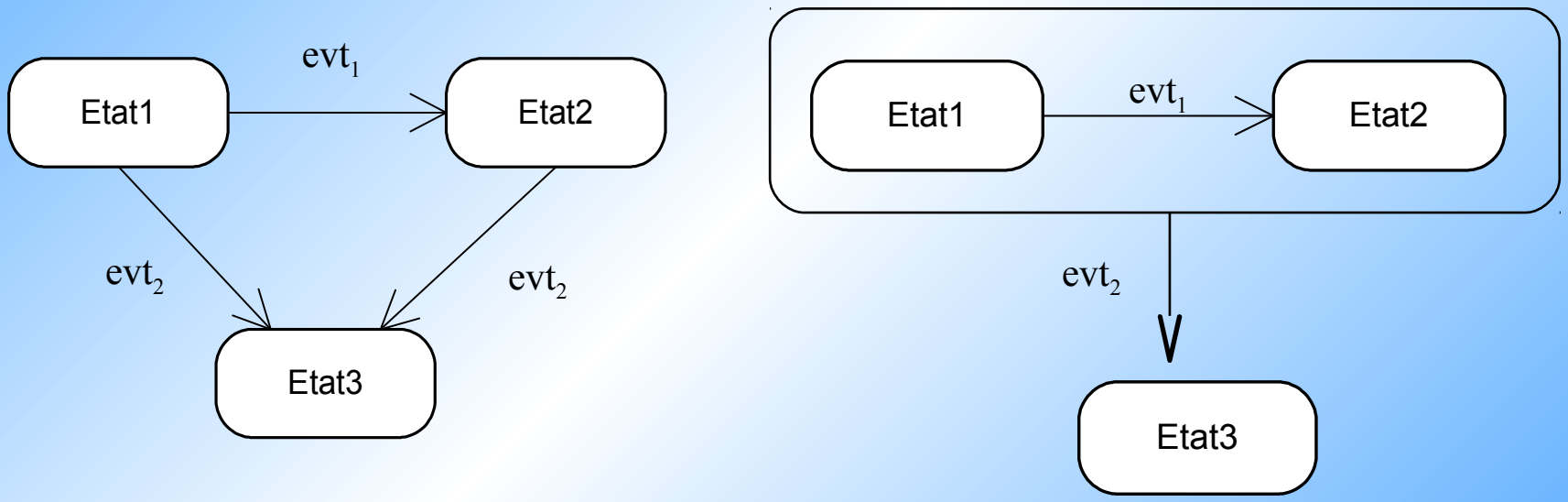
- Si un **événement** se produit et n'a **pas de transition** à partir de l'état courant, alors l'**événement** est **ignoré**.

Généralisation d'états

- Dans le cas d'un **comportement dynamique complexe**, les diagrammes d'états sur un niveau deviennent vite **illisibles**.
- Pour éviter ce problème, il est nécessaire de **structurer les diagrammes d'états**.
- Un **diagramme d'états imbriqué** est en fait une **généralisation d'états**.
- La **généralisation** permet d'organiser les états et les événements de façon hiérarchique en utilisant l'**héritage**.
- Les états dans le diagramme imbriqué sont tous des **raffinements** du diagramme de haut niveau.
- Un **super-état** est la généralisation de plusieurs sous-états.

Généralisation d'états

Exemple

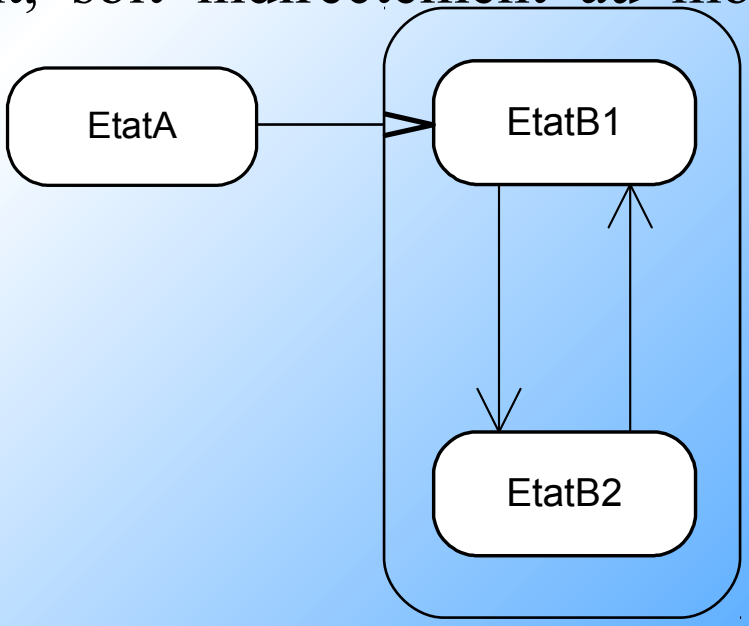


Généralisation d'états

Exemple

- L'état *B* est divisé en *2* sous-états B_1 et B_2.

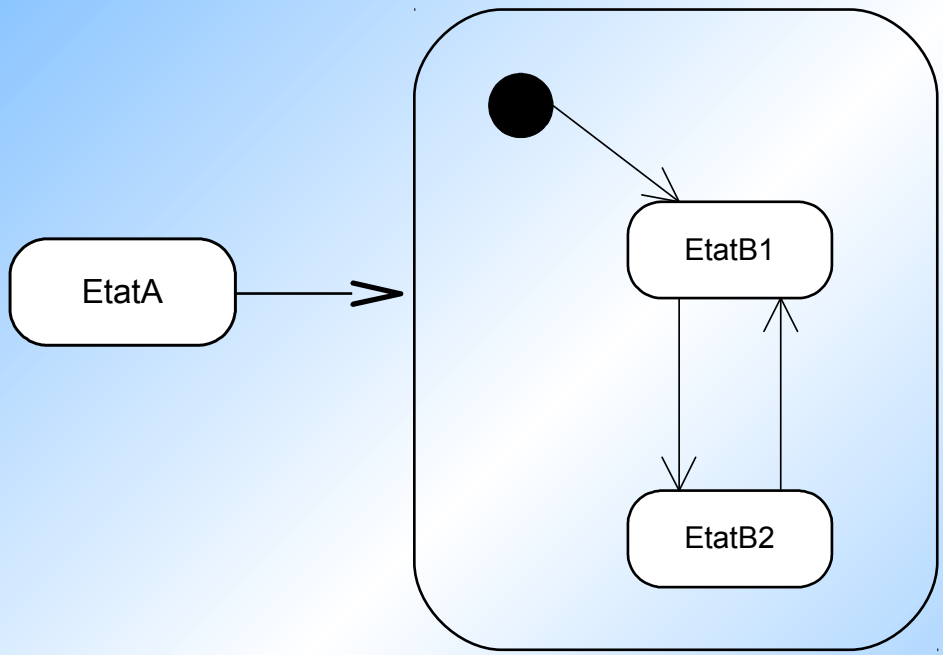
- La transition d'entrée dans l'état *B* doit être reportée sur un des sous-états soit directement, soit indirectement au moyen d'un état initial emboîté



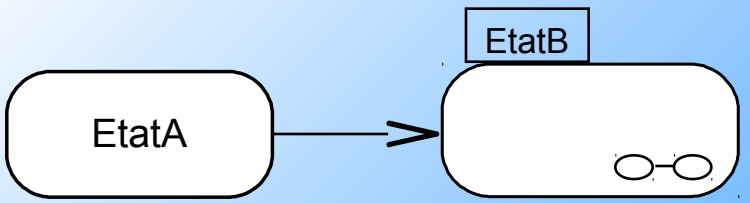
GÉNÉRALISATION D'ÉTATS

Exemple

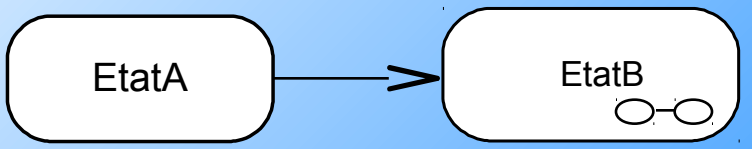
- Il est préférable de limiter les liens entre niveaux hiérarchiques en définissant systématiquement un état initial pour chaque niveau.



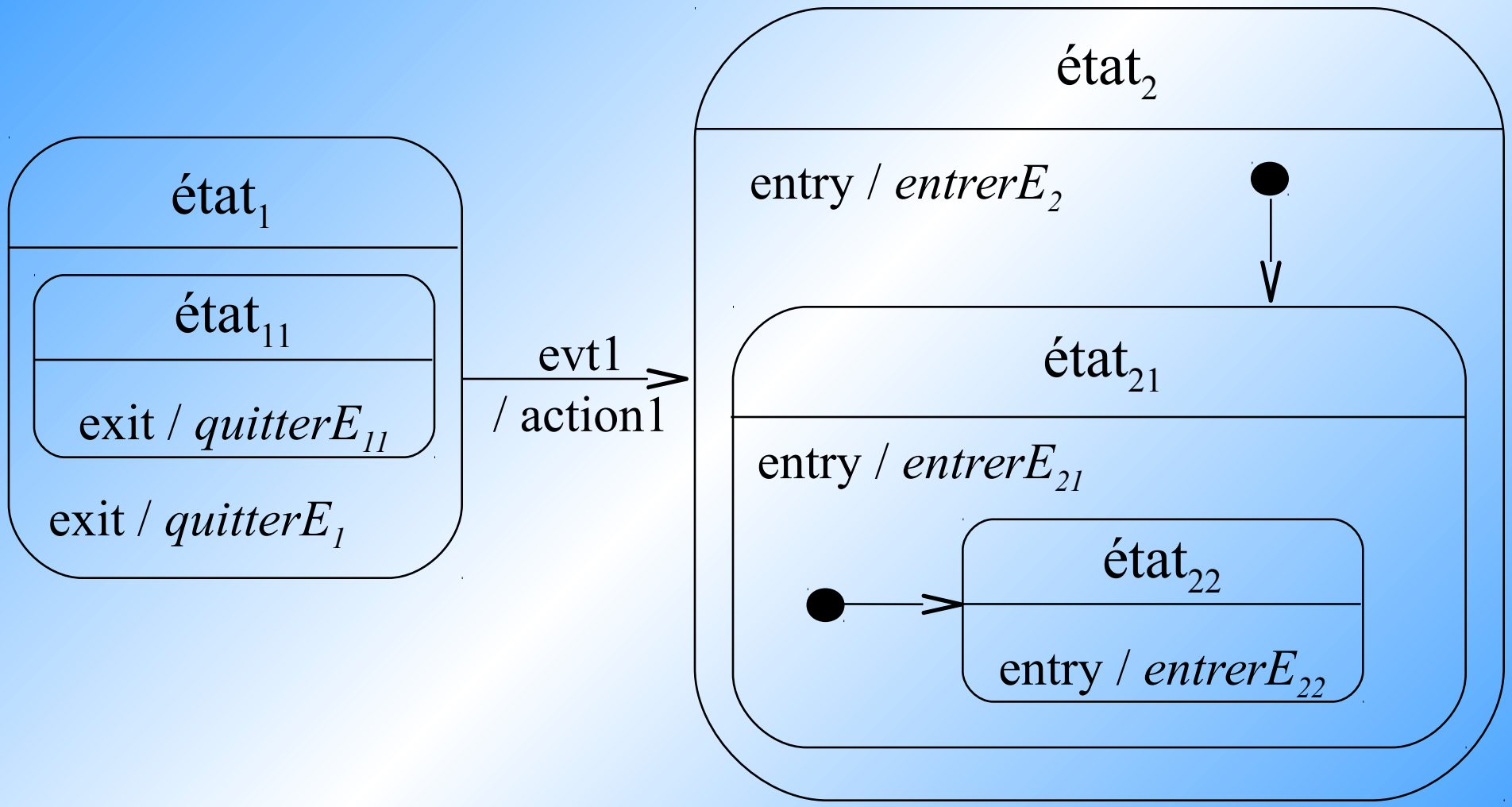
Notation



ou



EXERCICE



état$_{11}$, réception de l'événement *evt*$_1$

SOLUTION

- quitterE$_{11}$
- quitterE$_{1}$
- action$_{1}$
- entrerE$_{2}$
- entrerE$_{21}$
- entrerE$_{22}$

état$_{22}$

AGRÉGATION D'ÉTATS

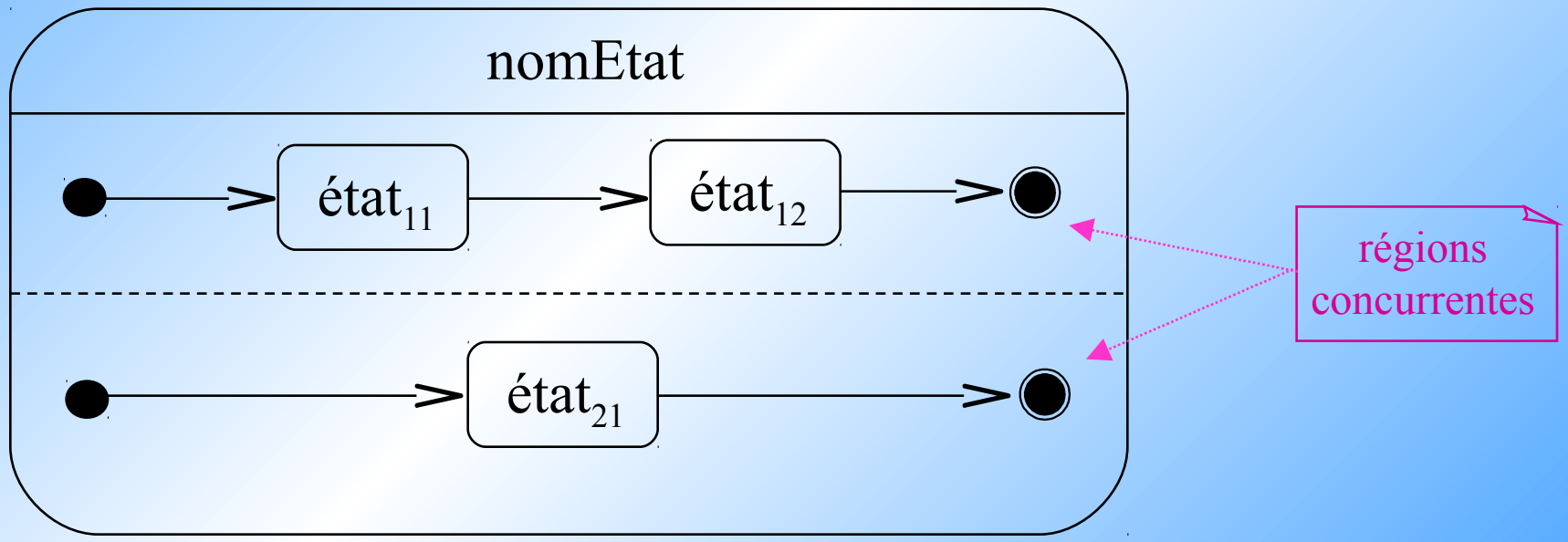
- L'**agrégation d'états** est la composition d'un état à partir de plusieurs autres états indépendants.

- L'objet doit être simultanément dans tous les états composant l'agrégation d'états.

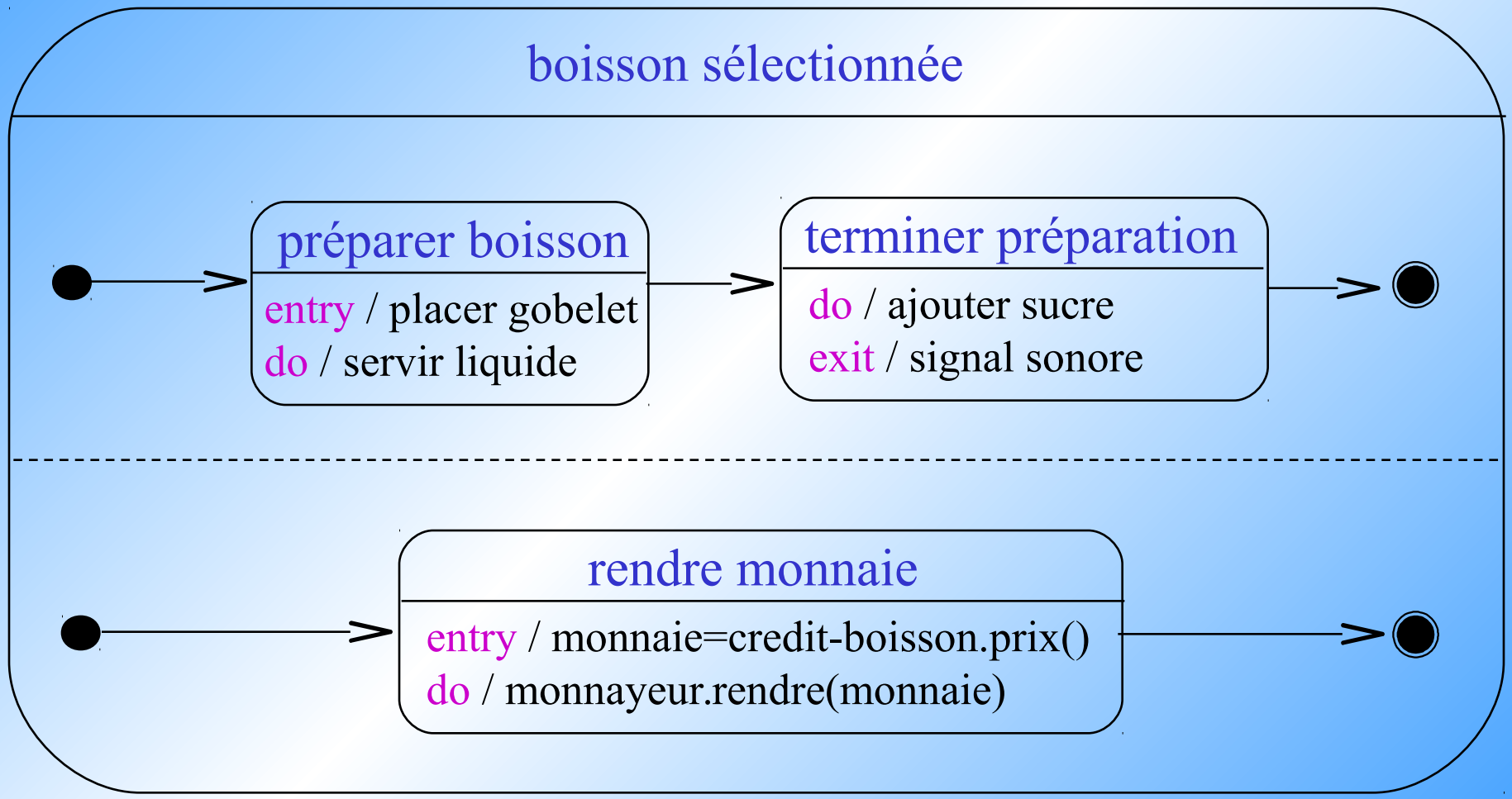
- L'agrégation d'états, tout comme la généralisation d'états, permet de **simplifier la représentation** des automates.
  - ✓ la **généralisation** simplifie par **factorisation**,
  - ✓ l'**agrégation** simplifie par **segmentation** de l'espace des états.

AGRÉGATION D'ÉTATS

- L'**agrégation d'états** permet de décrire efficacement les mécanismes concurrents. Cela permet de représenter des zones où l'action est réalisée par des **flots d'exécution parallèles**.

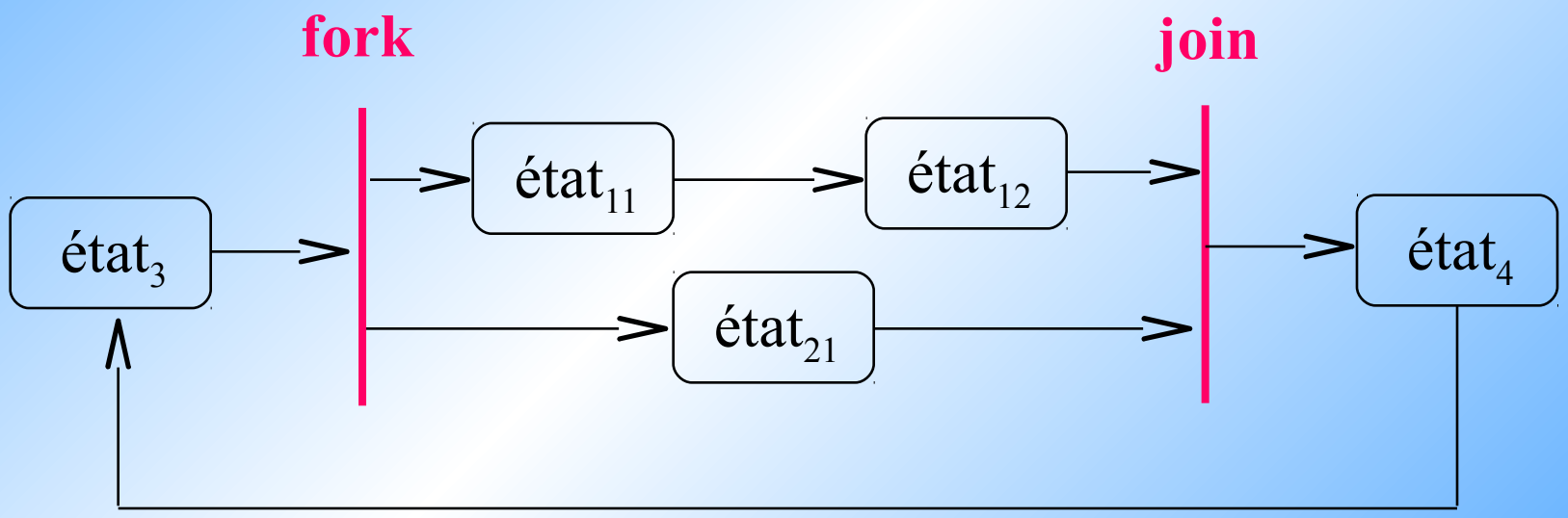


EXEMPLE


boisson sélectionnée
préparer boisson
entry / placer gobelet
do / servir liquide
terminer préparation
do / ajouter sucre
exit / signal sonore
rendre monnaie
entry / monnaie=credit-boisson.prix()
do / monnayeur.rendre(monnaie)

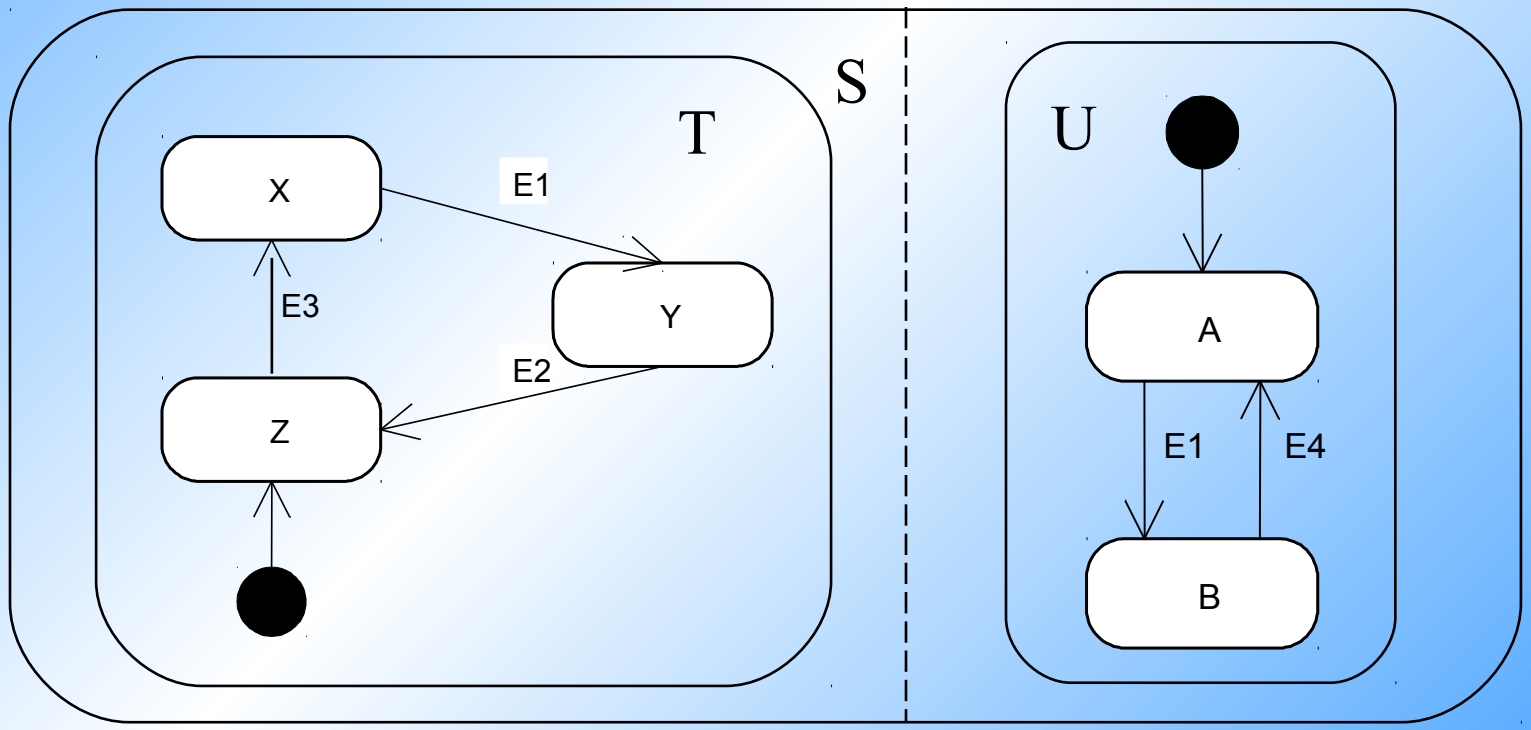
AGRÉGATION D'ÉTATS

- Il est également possible de représenter ce type de comportement au moyen de transitions concurrentes dites complexes.



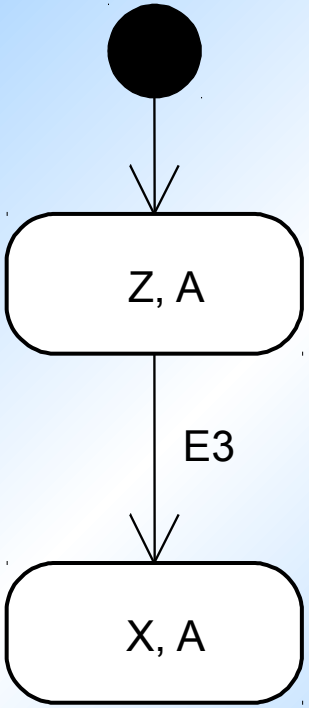
EXERCICE

- L'état *S* est un agrégat formé de *2* états indépendants *T* et *U*.
- *T* est composé des sous-états *X*, *Y* et *Z*.
- *U* est composé des sous-états *A* et *B*.



EXERCICE

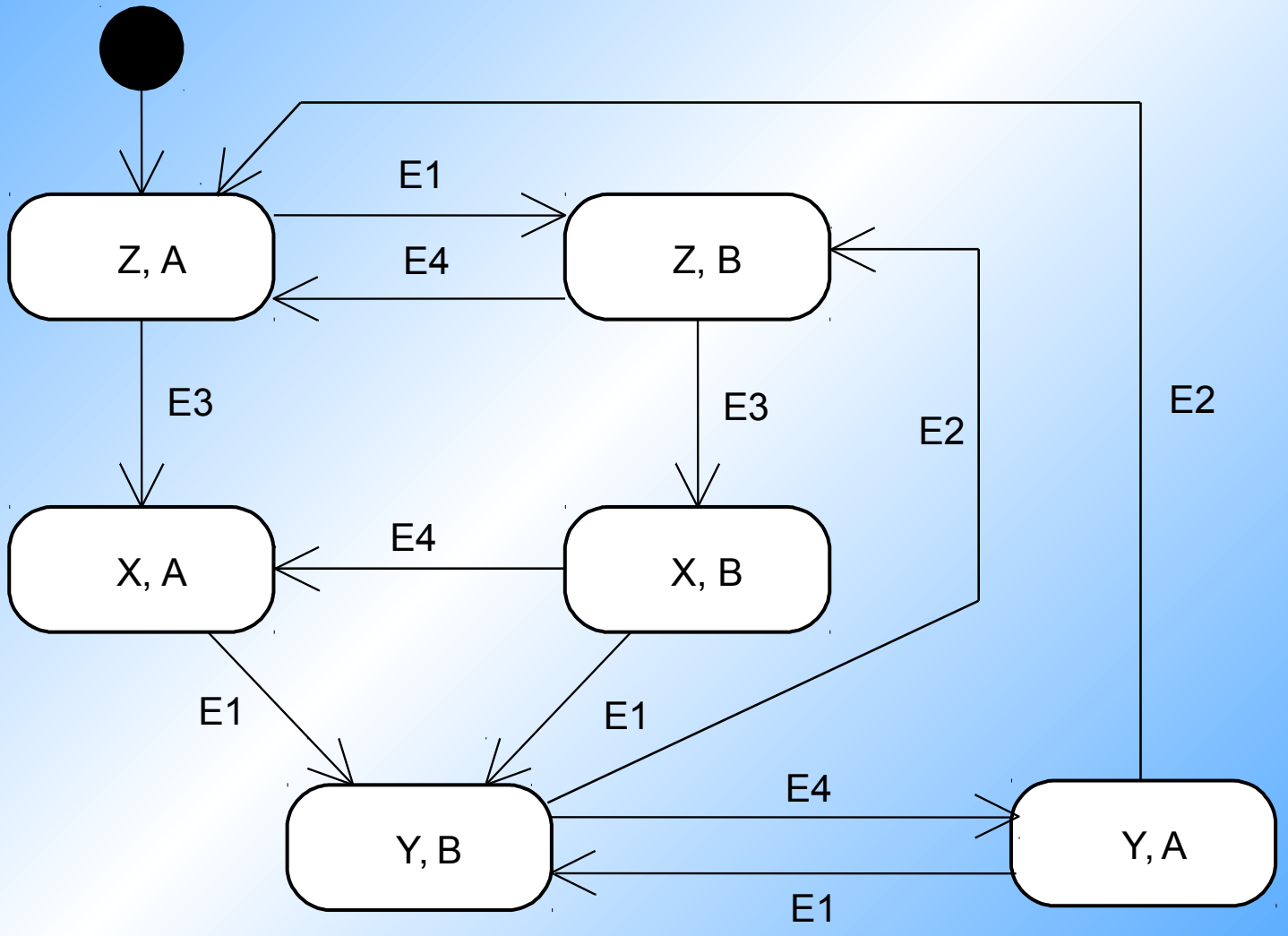
Sans agrégation d'états, chercher l'automate équivalent



A finir

6 états et *11* transitions

SOLUTION



POINTS DE DÉCISION

- Représentation des **alternatives pour le franchissement d'une transition**.

- Pseudo-états particuliers :

  - ✓ le point de jonction 

  - ✓ le point de choix 

POINT DE JONCTION

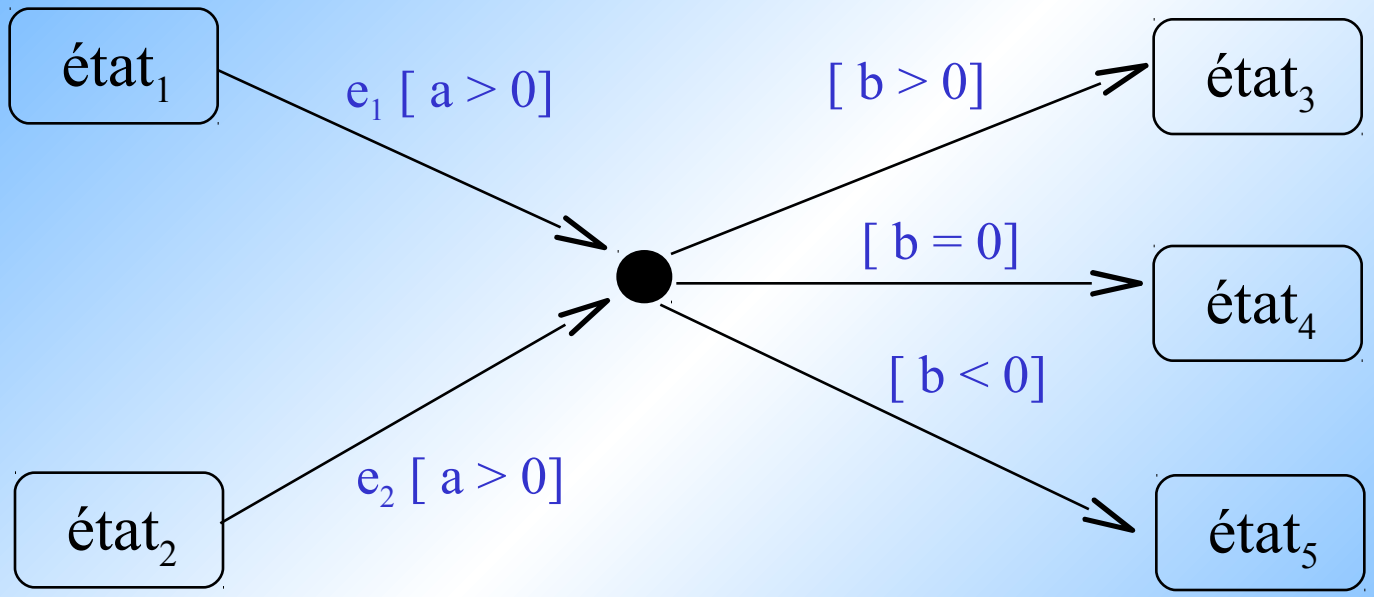
- Les **points de jonction** sont un artefact graphique qui permet de **partager des segments de transitions**.

- L'intérêt est de permettre une **notation plus compacte** et de rendre **plus visibles les chemins alternatifs**.

- Un **point de jonction** peut avoir **plusieurs segments** de transition **entrante** et **plusieurs segments** de transition **sortante**.

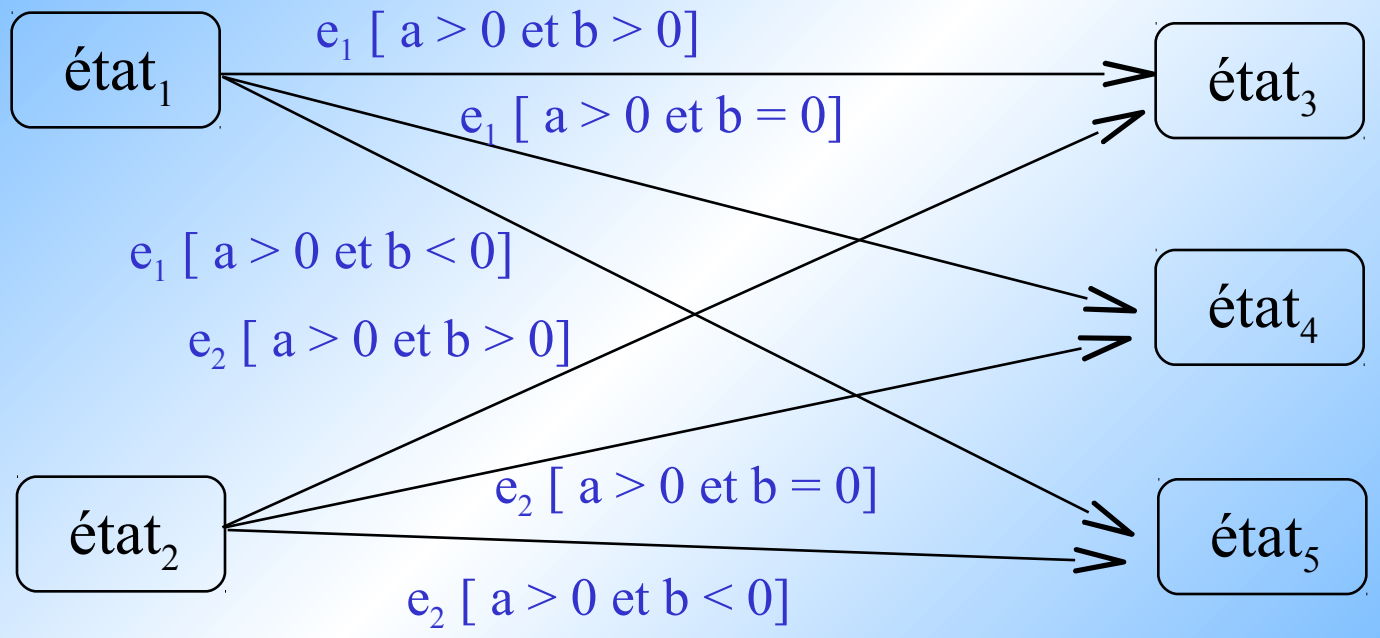
- Lorsqu'un chemin passant par un point de jonction est emprunté, toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.

EXERCICE



Sans point de jonction,
chercher l'automate équivalent

SOLUTION


état1
état2
état3
état4
état5
e1 [a > 0 et b > 0]
e1 [a > 0 et b = 0]
e1 [a > 0 et b < 0]
e2 [a > 0 et b > 0]
e2 [a > 0 et b = 0]
e2 [a > 0 et b < 0]

POINT DE CHOIX

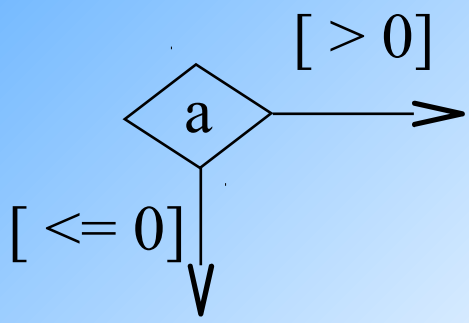
- Un **point de choix ou décision** possède **une entrée** et **au moins deux sorties**.

- Contrairement aux points de jonction, les **points de choix** dits *"dynamiques"*, sont **évalués au moment où ils sont atteints**.

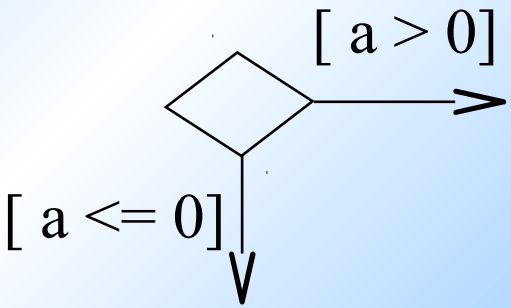
- Si quand le point de choix est atteint, aucun segment en aval n'est franchissable, le modèle est dit mal formé.

- Au contraire, si plusieurs segments sont franchissables, l'une d'entre elles est choisie aléatoirement.

POINT DE CHOIX

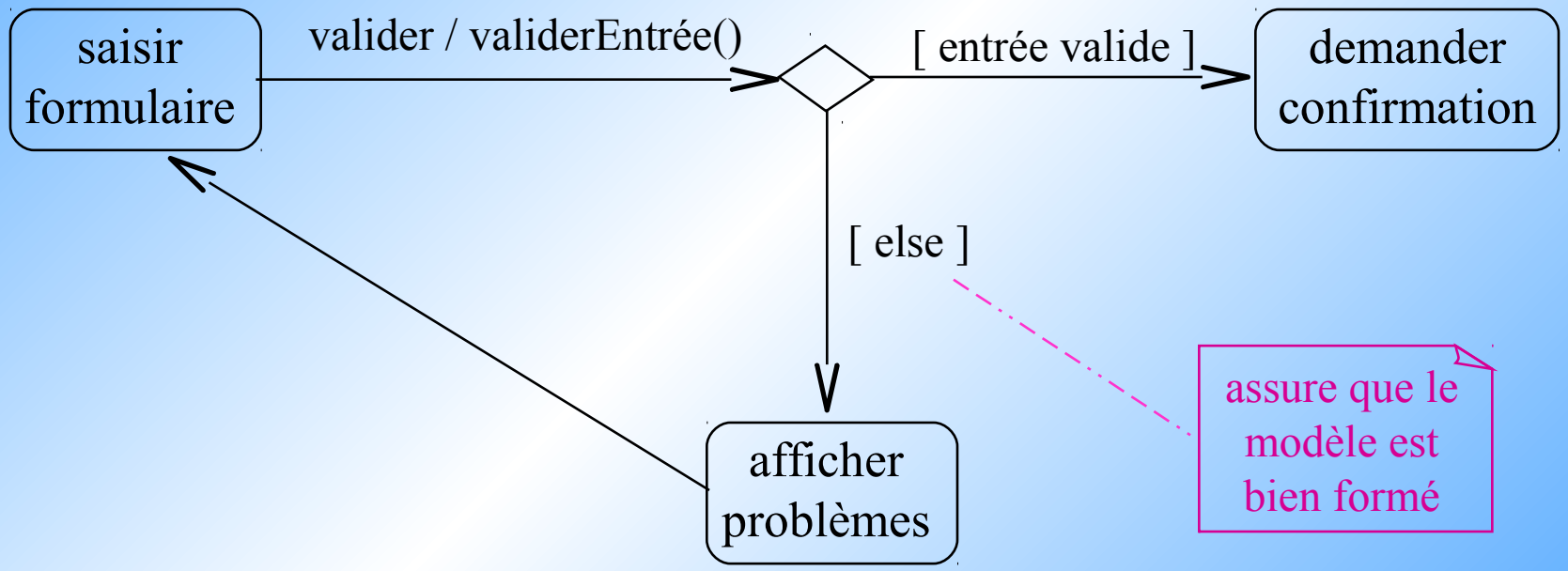


ou

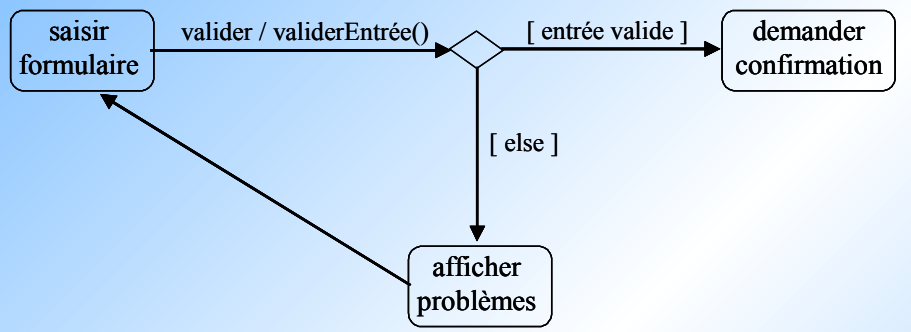


POINT DE CHOIX

Exemple



POINT DE CHOIX



- L'utilisateur remplit un formulaire. Il valide son formulaire en appuyant sur le bouton *valider*.

- *validerEntrée()* effectue une vérification de la cohérence des données.

- Si les informations paraissent correctes, on lui demande de confirmer, sinon on affiche les erreurs détectées et il doit remplir de nouveau le formulaire.

HISTORIQUE

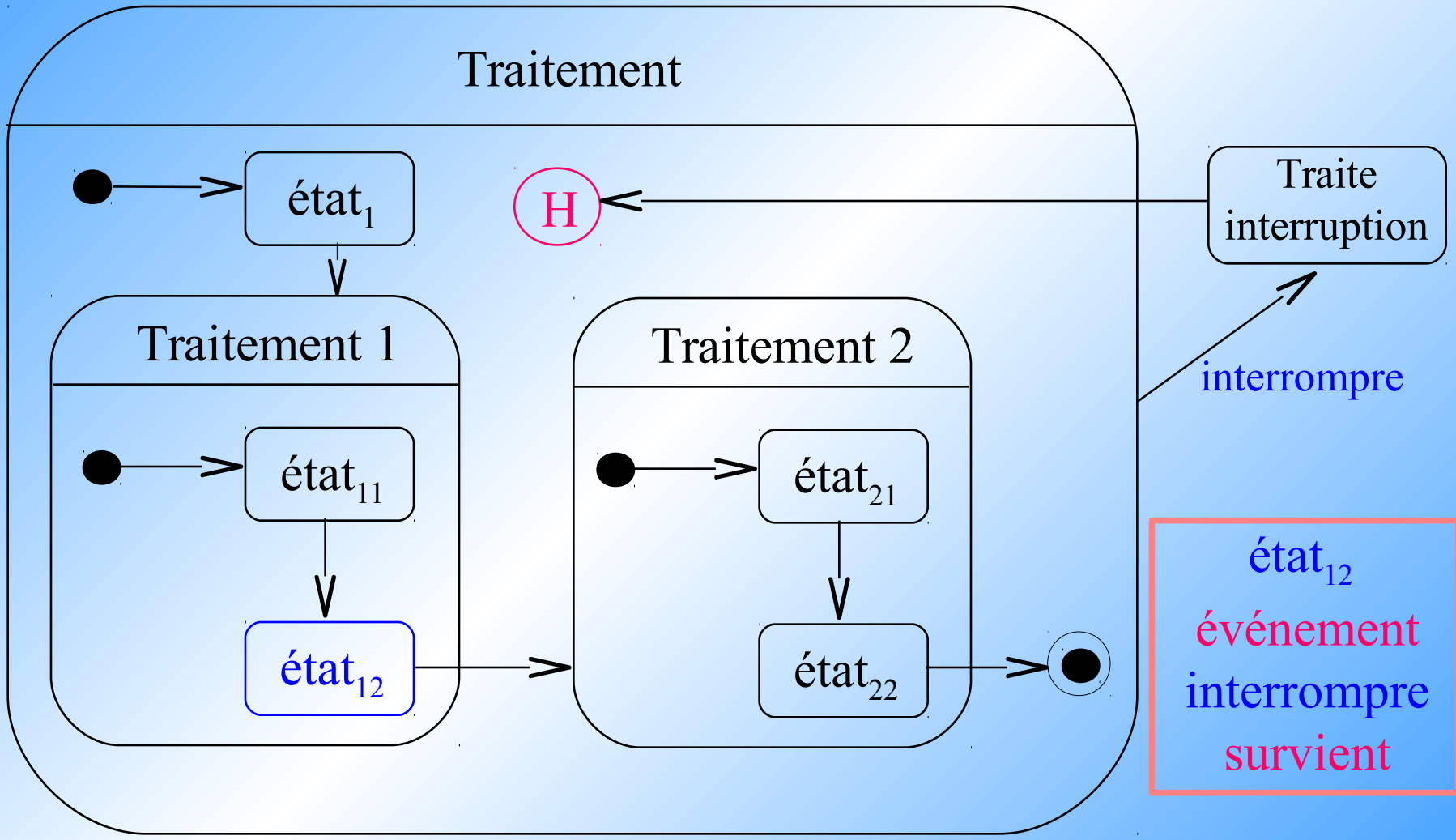
- Une transition ayant pour cible le **pseudo-état historique**, noté par un cercle contenant un H, est équivalente à une transition qui a pour **cible le dernier état visité dans la région** contenant le H.

historique de surface

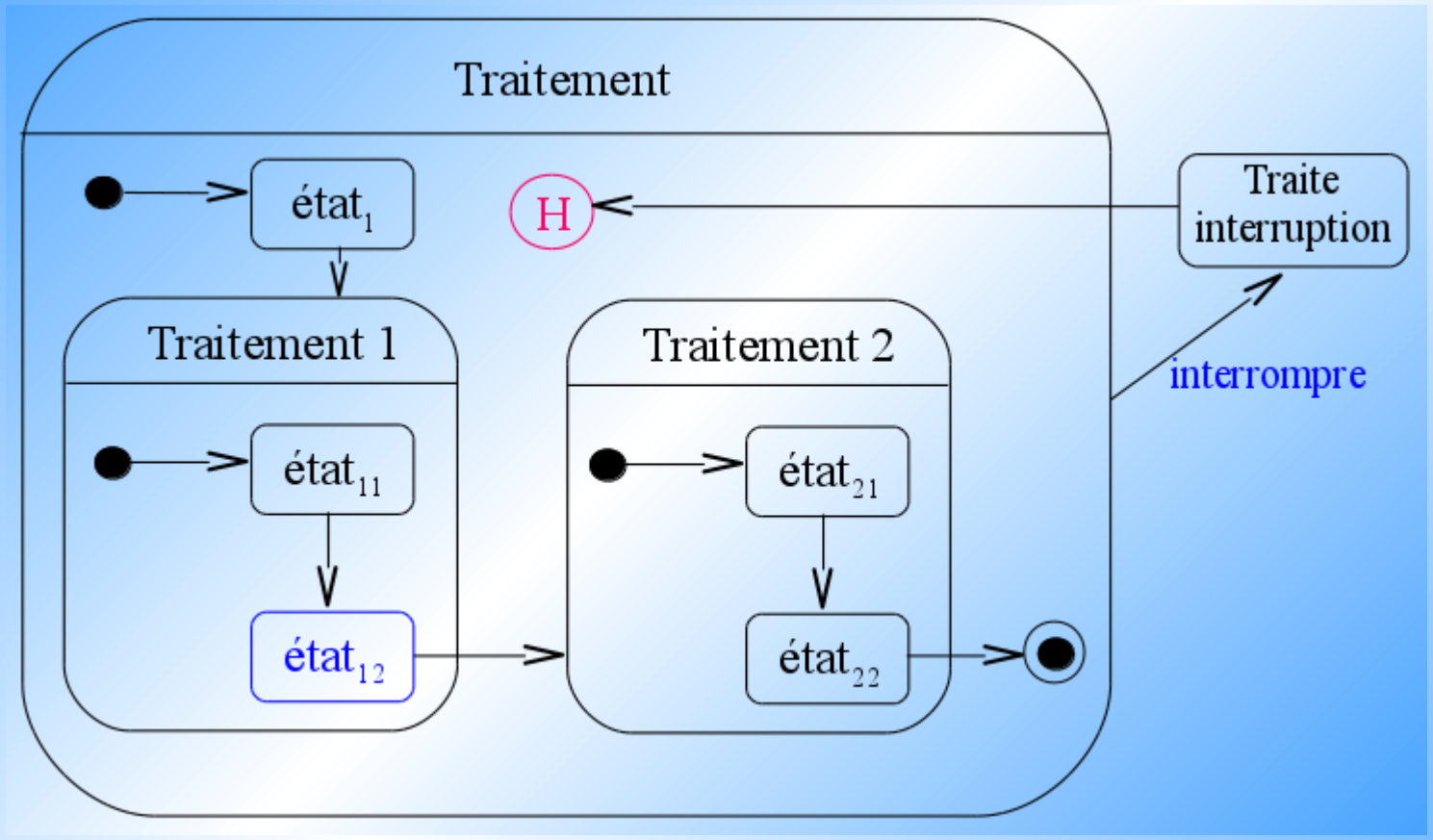
- Une transition ayant pour cible le **pseudo-état historique profond**, noté par un cercle contenant un H*, permet d'atteindre le **dernier état visité dans la région**, **quel que soit son niveau d'imbrication**, alors que le pseudo-état H limite l'accès aux états de son imbrication dans la région.

historique profond

EXERCICE


Traitement
état1
H
Traite interruption
Traitement 1
Traitement 2
interrompre
état11
état21
état12
état22
état12
événement
interrompre
survient

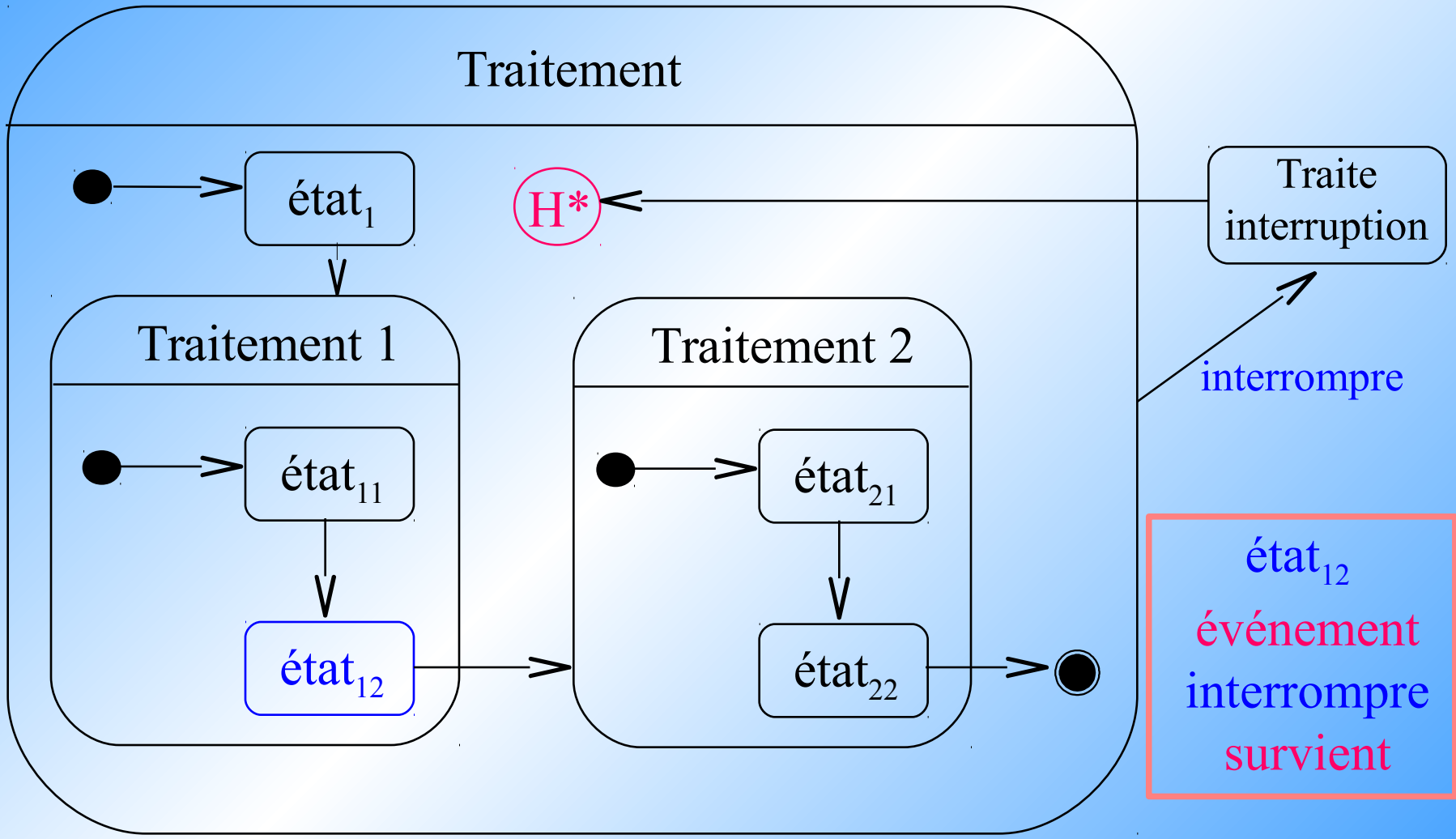
SOLUTION



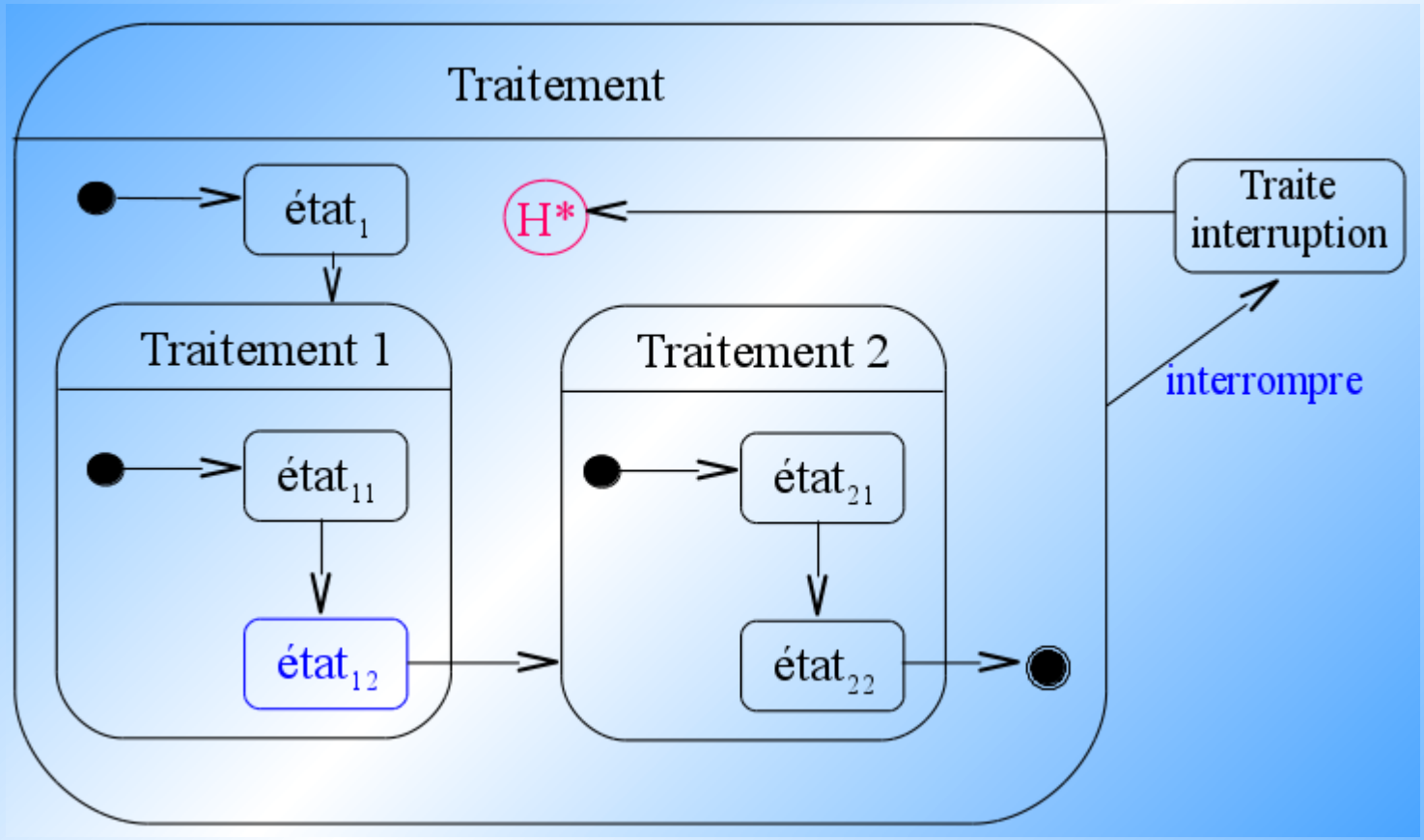
état$_{12}$

interrompre ⇨ état$_{11}$

EXERCICE


Traitement
état1
H*
Traite interruption
Traitement 1
Traitement 2
interrompre
état11
état21
état12
état22
état12
événement
interrompre
survient

SOLUTION



état$_{12}$

interrompre ⇒ état$_{12}$

ÉTUDE DE CAS

Système simplifié de Publiphone à pièces

1- Le prix minimal d'une communication interurbaine est de *1* € .
2- Après l'introduction de la monnaie, l'utilisateur a deux minutes pour composer son numéro (*ce délai est décompté par le standard*).
3- La ligne peut être libre ou occupée.
4- Le correspondant peut raccrocher le premier.
5- Le Publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps (*UT*) générée par le standard.
6- On peut ajouter des pièces à tout moment.
7- Lors du raccrochage, le solde de monnaie est rendu.

ÉTUDE DE CAS

1- Identifier les acteurs

2- Construire le diagramme de contexte statique

3- Identifier les cas d'utilisation

4- Construire un diagramme de séquence système.

5- Construire le diagramme de contexte dynamique.

6- Élaborer le diagramme d'états du Publiphone.

ÉTUDE DE CAS

Identification des acteurs

Quelles sont les entités externes qui interagissent directement avec le Publiphone ?

SOLUTION

Si on procède à une analyse linguistique de l'énoncé, on obtient les *4* candidats suivants :

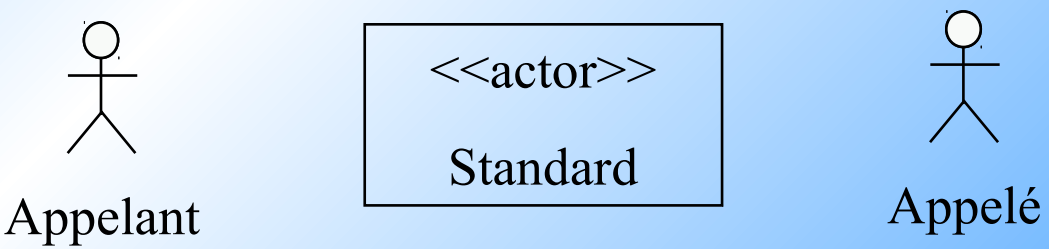
- Utilisateur
- Standard
- Correspondant
- Appelé

SOLUTION

- Le standard est un acteur (*non-humain*) connecté directement au système.

- La seule difficulté concerne les acteurs humains : utilisateur, correspondant et appelé.

Comme les deux derniers sont des synonymes, on peut garder le mot *appelé* et renommer, par souci de symétrie, utilisateur en *appelant*.



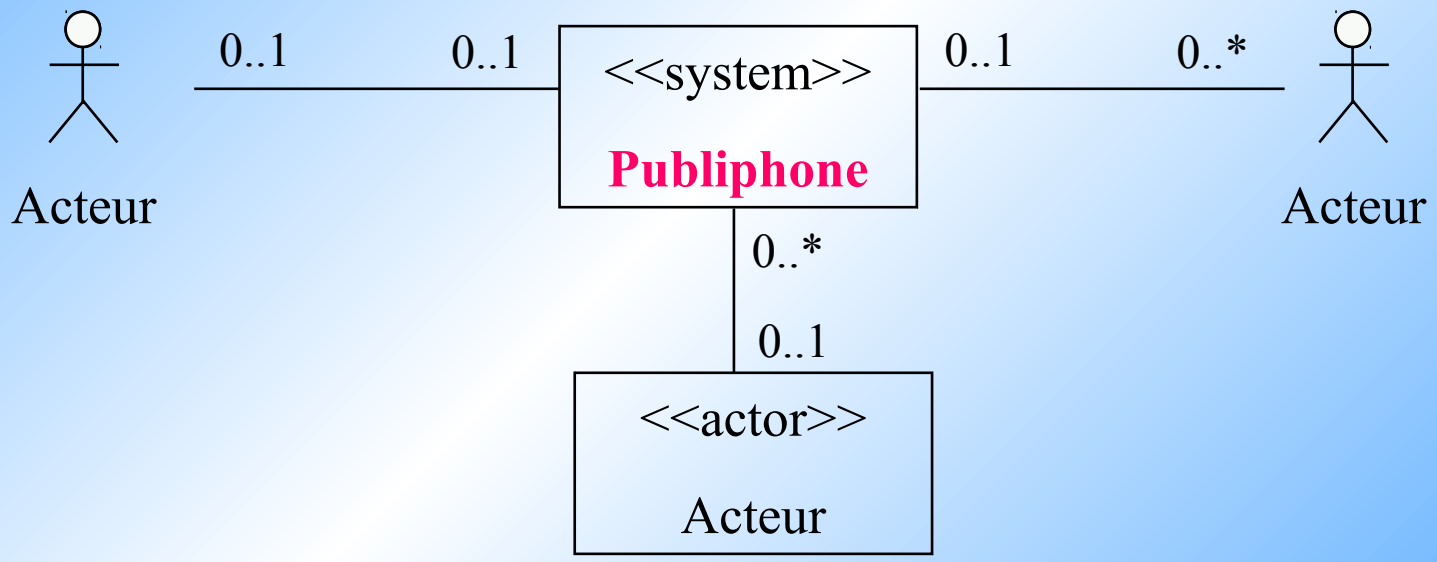
ÉTUDE DE CAS

Construire le diagramme de contexte statique

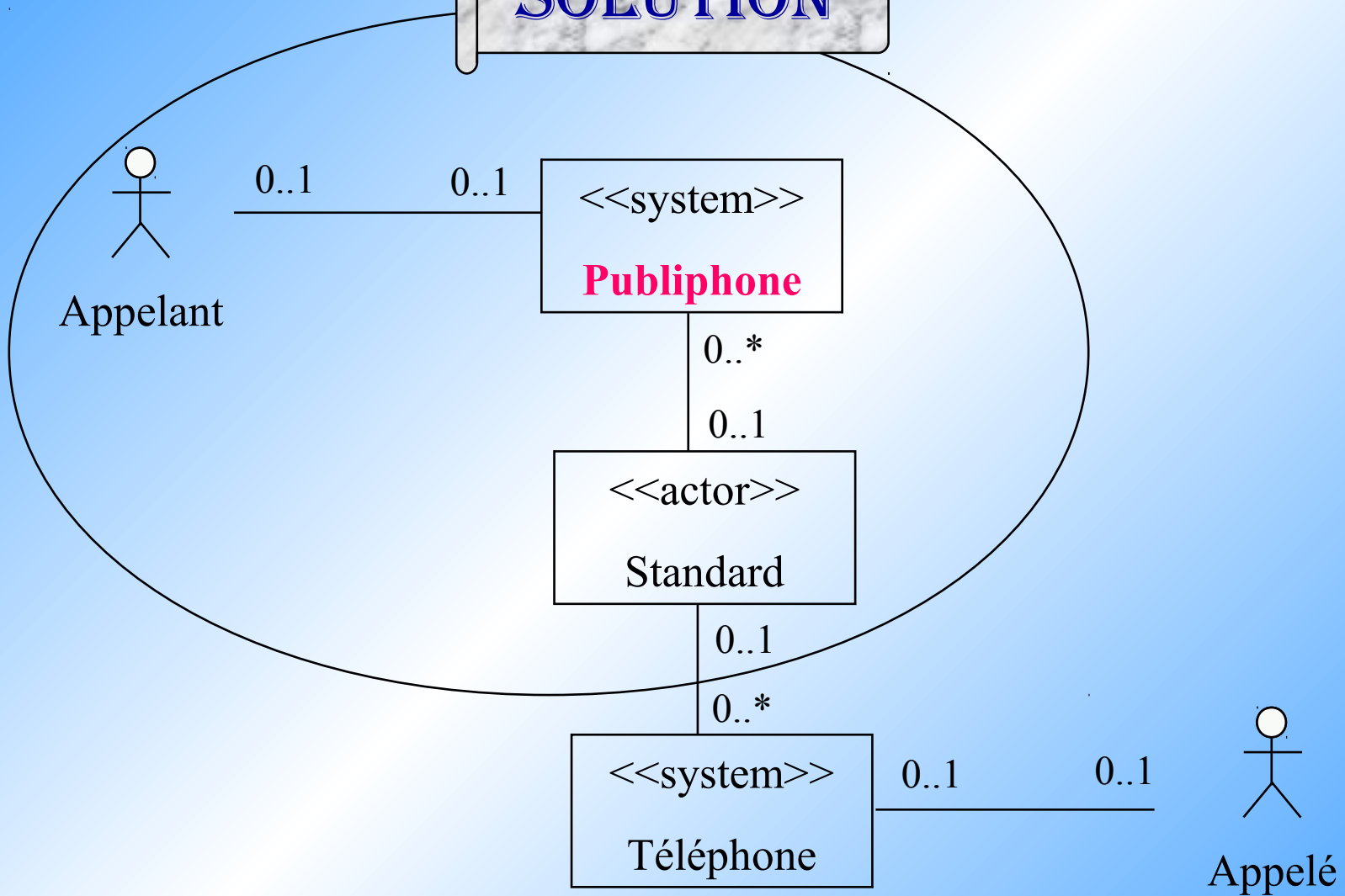
Liens acteurs - système
avec les cardinalités

SOLUTION

Exemple



SOLUTION


Appelant
0..1
0..1
<<system>>
Publiphone
0..*
0..1
<<actor>>
Standard
0..1
0..*
<<system>>
Téléphone
0..1
0..1
Appelé

SOLUTION

Comment les acteurs utilisent-ils le Publiphone ?

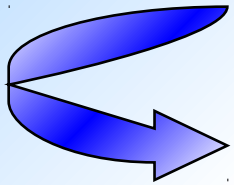
- La seule utilisation vraiment intéressante dans notre contexte est celle de l'appelant qui téléphone à l'appelé.

- Le standard sert en fait d'intermédiaire entre les deux. Si on affine encore notre analyse, on s'aperçoit rapidement que l'appelé n'interagit pas directement avec le Publiphone : il est complètement masqué par le standard.

SOLUTION

- On voit que l'appelant communique avec l'appelé par le biais de trois systèmes connectés : le Publiphone, le standard et le téléphone de l'appelé.

- L'appelé est donc un acteur indirect par rapport au Publiphone.



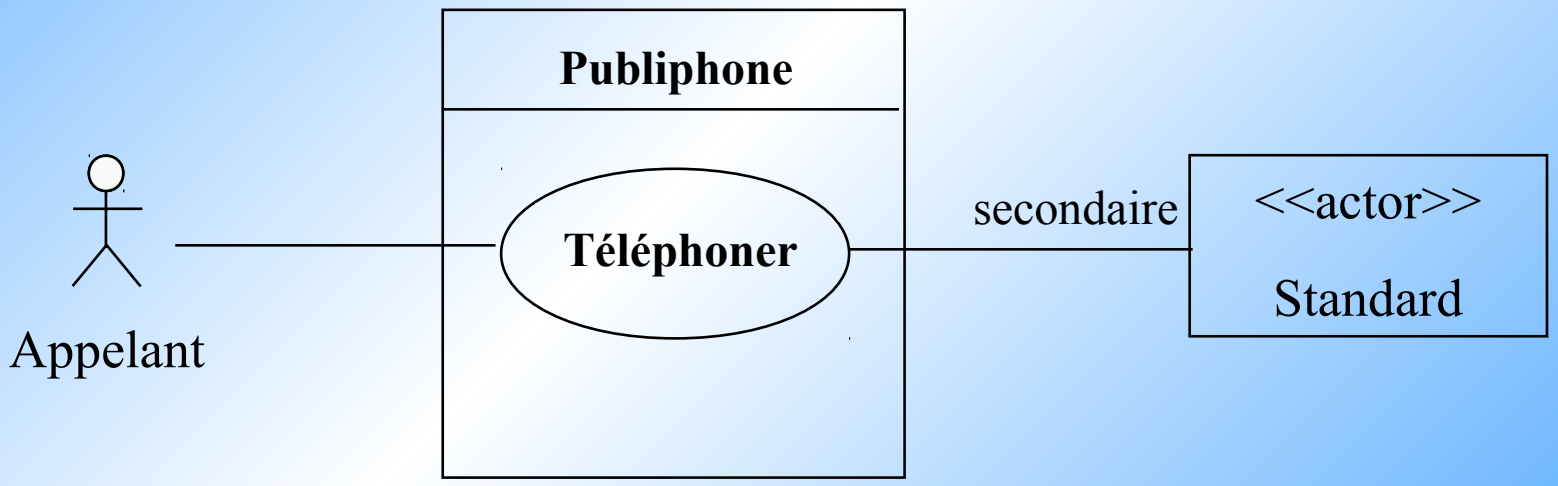
On ne le retiendra pas pour notre diagramme de cas d'utilisation.

ÉTUDE DE CAS

Faire le diagramme de cas d'utilisation

SOLUTION

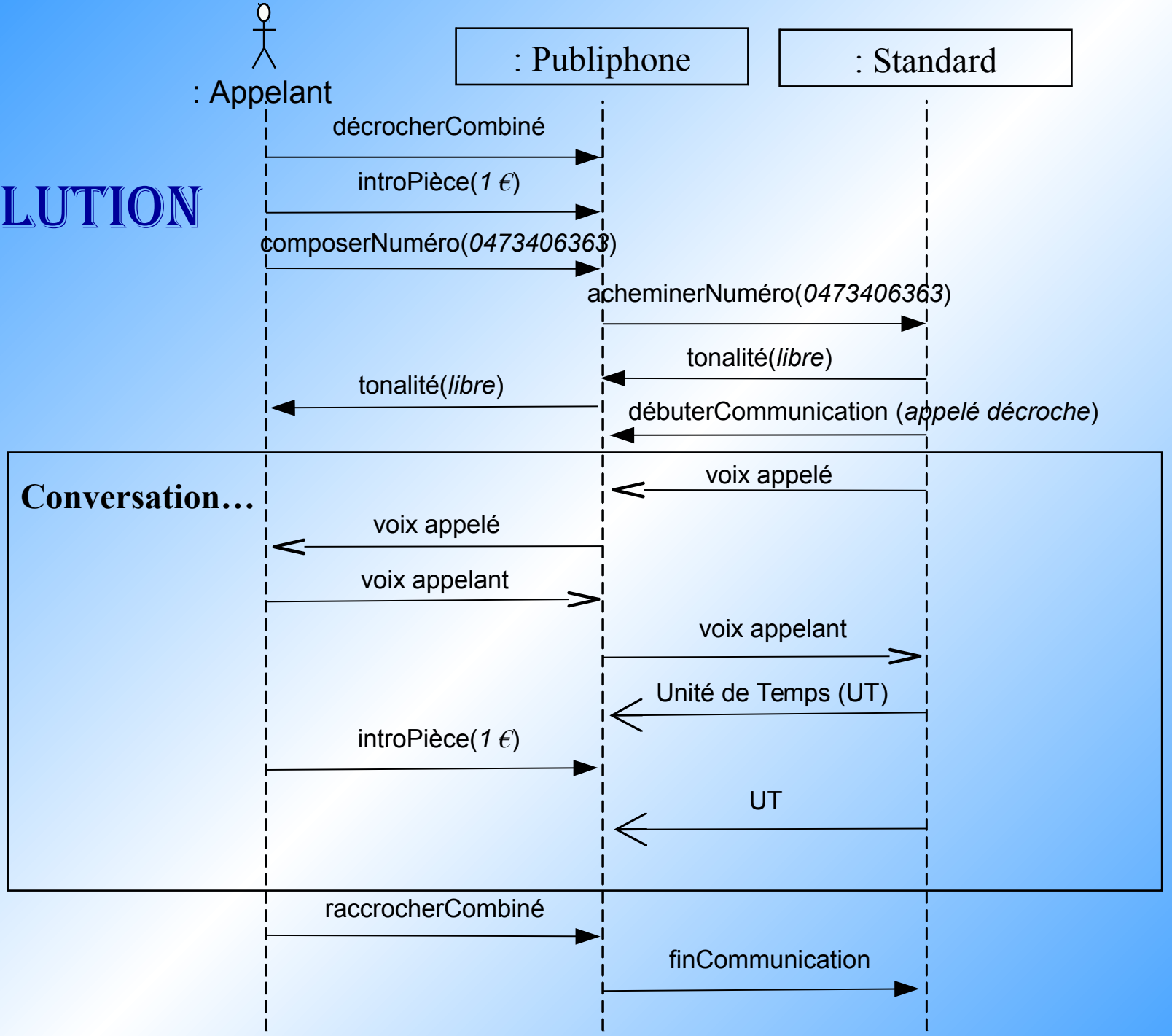
Diagramme de cas d'utilisation



ÉTUDE DE CAS

Faire le diagramme de séquence système qui décrive le scénario nominal du cas d'utilisation Téléphoner

SOLUTION

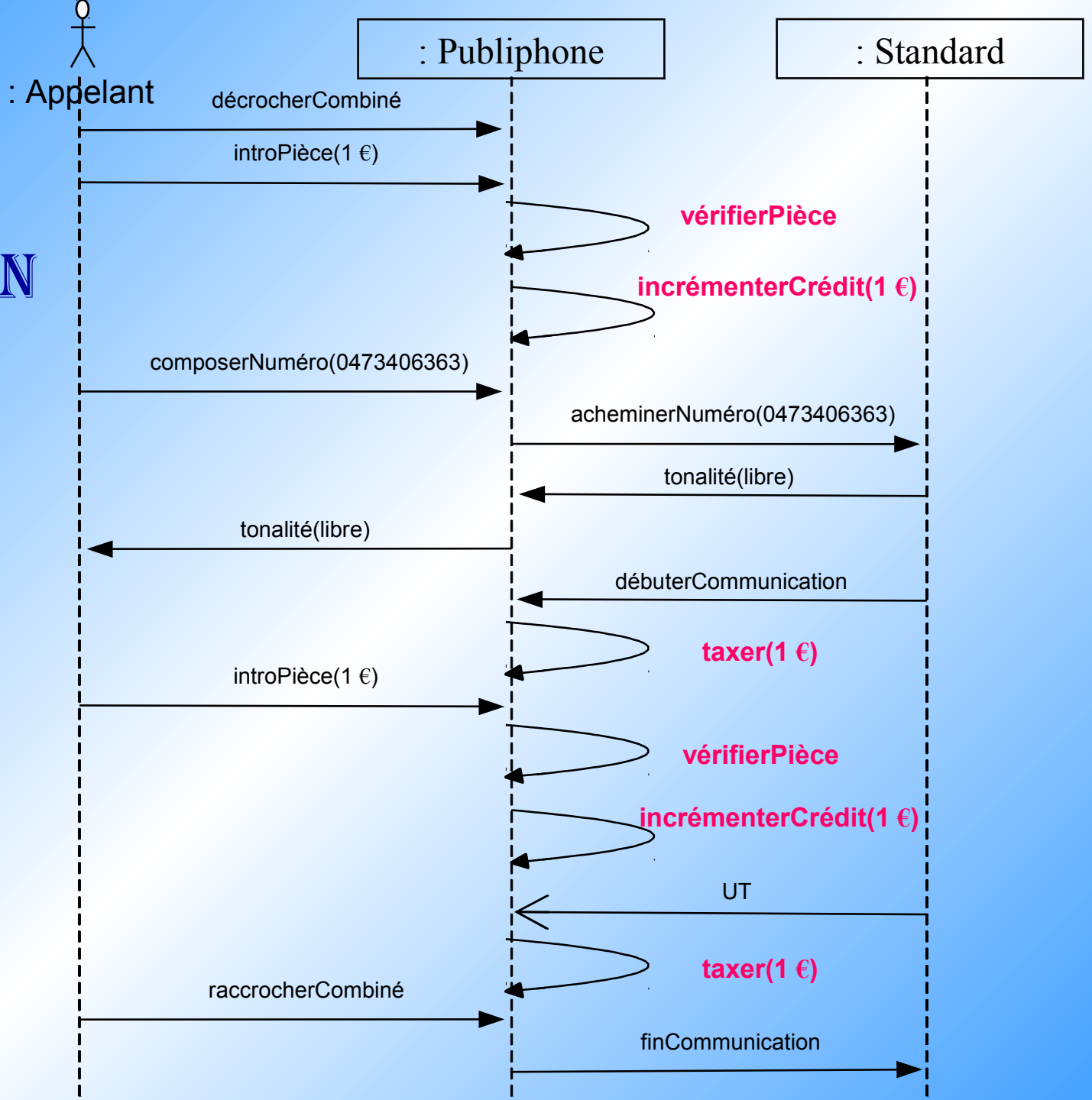

: Appelant
: Publiphone
: Standard
décrocherCombiné
introPièce(1 €)
composerNuméro(0473406363)
acheminerNuméro(0473406363)
tonalité(libre)
tonalité(libre)
débuterCommunication (appelé décroche)
Conversation…
voix appelé
voix appelé
voix appelant
voix appelant
Unité de Temps (UT)
introPièce(1 €)
UT
raccrocherCombiné
finCommunication

ÉTUDE DE CAS

Enrichissez le diagramme de séquence système avec des activités internes intéressantes.

- Vérification des pièces
- Gestion du solde de l'appelant :
  incrémentation
  décrémentation (*taxer*)

SOLUTION


: Appelant
: Publiphone
: Standard
décrocherCombiné
introPièce(1 €)
vérifierPièce
incrémenterCrédit(1 €)
composerNuméro(0473406363)
acheminerNuméro(0473406363)
tonalité(libre)
tonalité(libre)
débuterCommunication
taxer(1 €)
introPièce(1 €)
vérifierPièce
incrémenterCrédit(1 €)
UT
taxer(1 €)
raccrocherCombiné
finCommunication

SOLUTION

Activités internes importantes du Publiphone :

- La vérification des pièces
- La gestion du solde de l'appelant :
  - ✓ incrémentation lors de l'introduction de pièces
  - ✓ décrémentation lors du début de communication et à chaque UT.

ÉTUDE DE CAS

- Afin de préparer le diagramme d'états, on va répertorier l'ensemble des messages qui sont émis et surtout ceux qui sont reçus par le Publiphone :

✓ Les messages reçus vont devenir des événements qui déclenchent des transitions entre états et

✓ Les messages émis vont donner lieu à des actions sur les transitions.

ÉTUDE DE CAS

Représentation graphique de l'ensemble des messages échangés par le système avec ses acteurs sous la forme d'un diagramme appelé ***diagramme de contexte dynamique***.

DIAGRAMME CONTEXTE DYNAMIQUE

Représentation graphique du contexte dynamique

Utilisez un diagramme de collaboration de la façon suivante :

- le système étudié est représenté par un objet au centre du diagramme ;

- cet objet central est entouré par une instance de chaque acteur ;

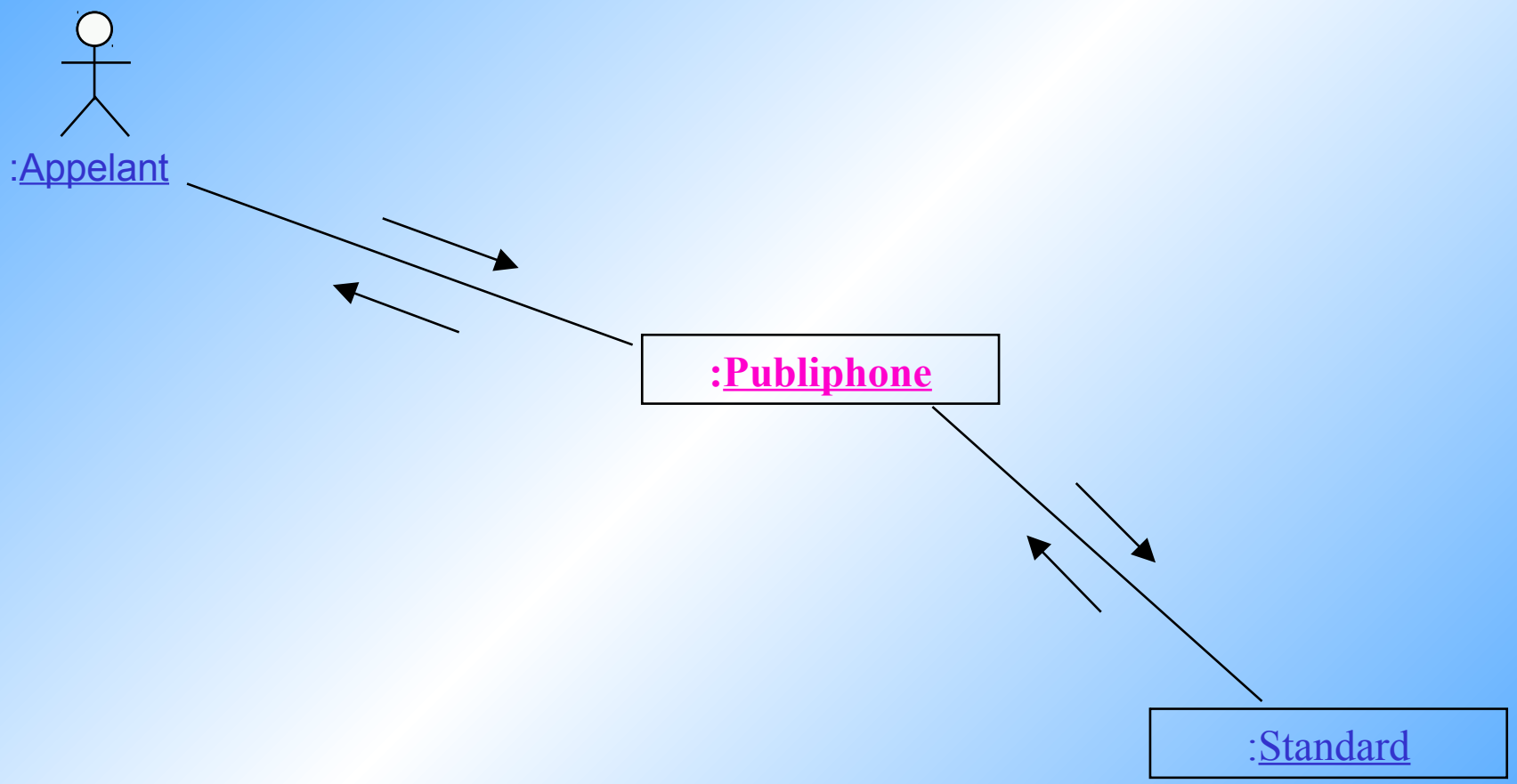
- un lien relie le système à chacun des acteurs ;

- sur chaque lien sont répertoriés tous les messages en entrée et en sortie du système, sans numérotation.

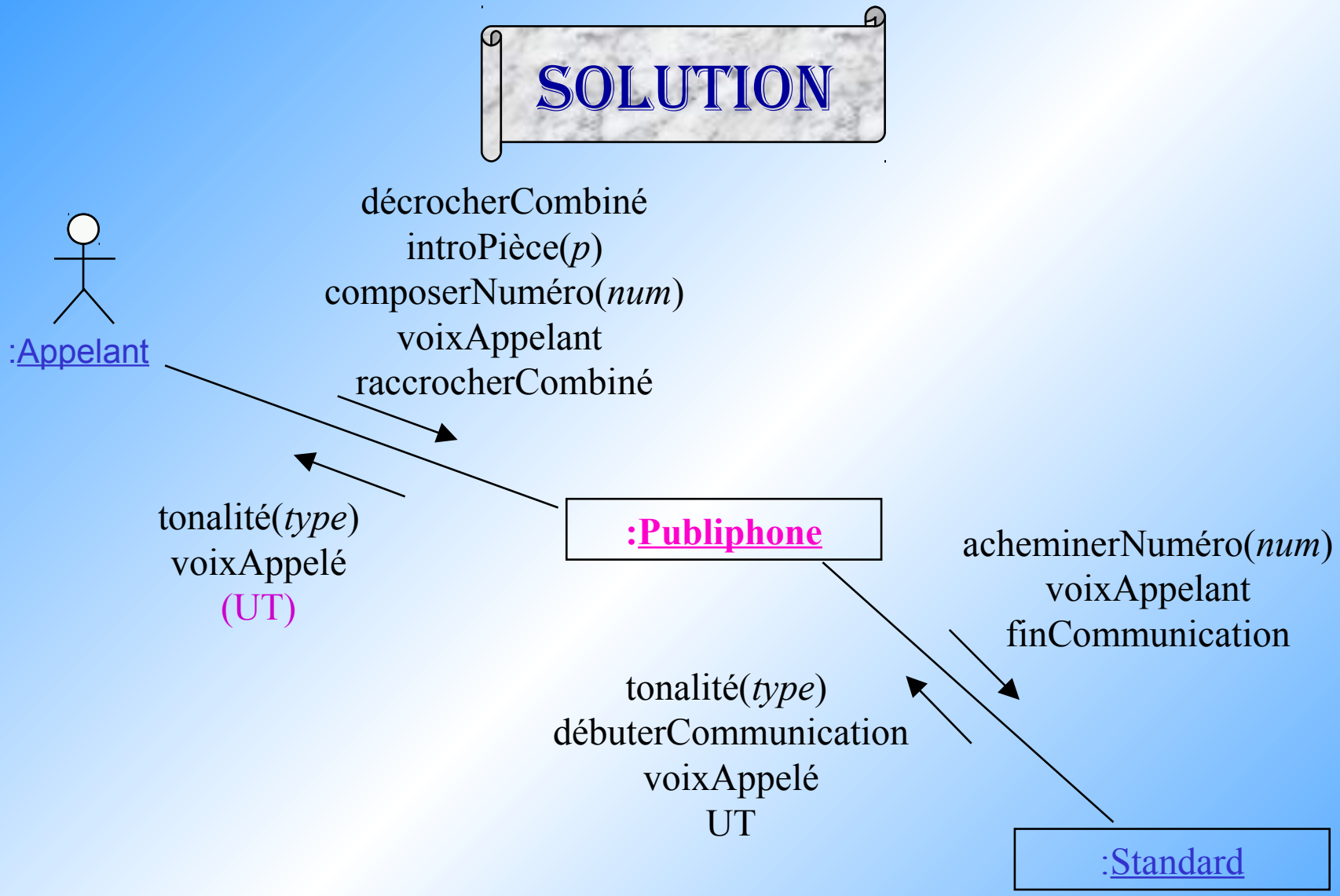
ÉTUDE DE CAS

Réaliser le diagramme de contexte dynamique du Publiphone

SOLUTION



SOLUTION


décrocherCombiné
introPièce(*p*)
composerNuméro(*num*)
voixAppelant
raccrocherCombiné
:Appelant
tonalité(*type*)
voixAppelé
(UT)
:Publiphone
acheminerNuméro(*num*)
voixAppelant
finCommunication
tonalité(*type*)
débuterCommunication
voixAppelé
UT
:Standard

SOLUTION

- Généralisation des messages échangés entre le système et ses acteurs en ajoutant des paramètres :

✓ *introPiece(1 €)* devient un message paramétré *introPiece(p)*

✓ *composerNumero(0473406363)* devient *composerNumero(num)*

✓ *tonalité(libre)* devient *tonalité(type)*
(*pour prendre en compte l'occupation de la ligne*)

ÉTUDE DE CAS

D'autres messages peuvent être envisagés entre le Publiphone et ses acteurs :

- s'il reste des pièces inutilisées quand l'appelant raccroche, le Publiphone les lui rend (*rendrePièces*) ;

- après l'introduction de la somme minimale de *1* €, le Publiphone transmet un message au standard pour le décompte du délai de *2* minutes (*timerNumérotation*) ;

- si le délai de *2* minutes est dépassé, le standard informe le Publiphone (*timeoutNumérotation*) ;

ÉTUDE DE CAS

D'autres messages peuvent être envisagés entre le Publiphone et ses acteurs :

- si le numéro composé est valide, le standard le détecte (*numéroValide*) ;

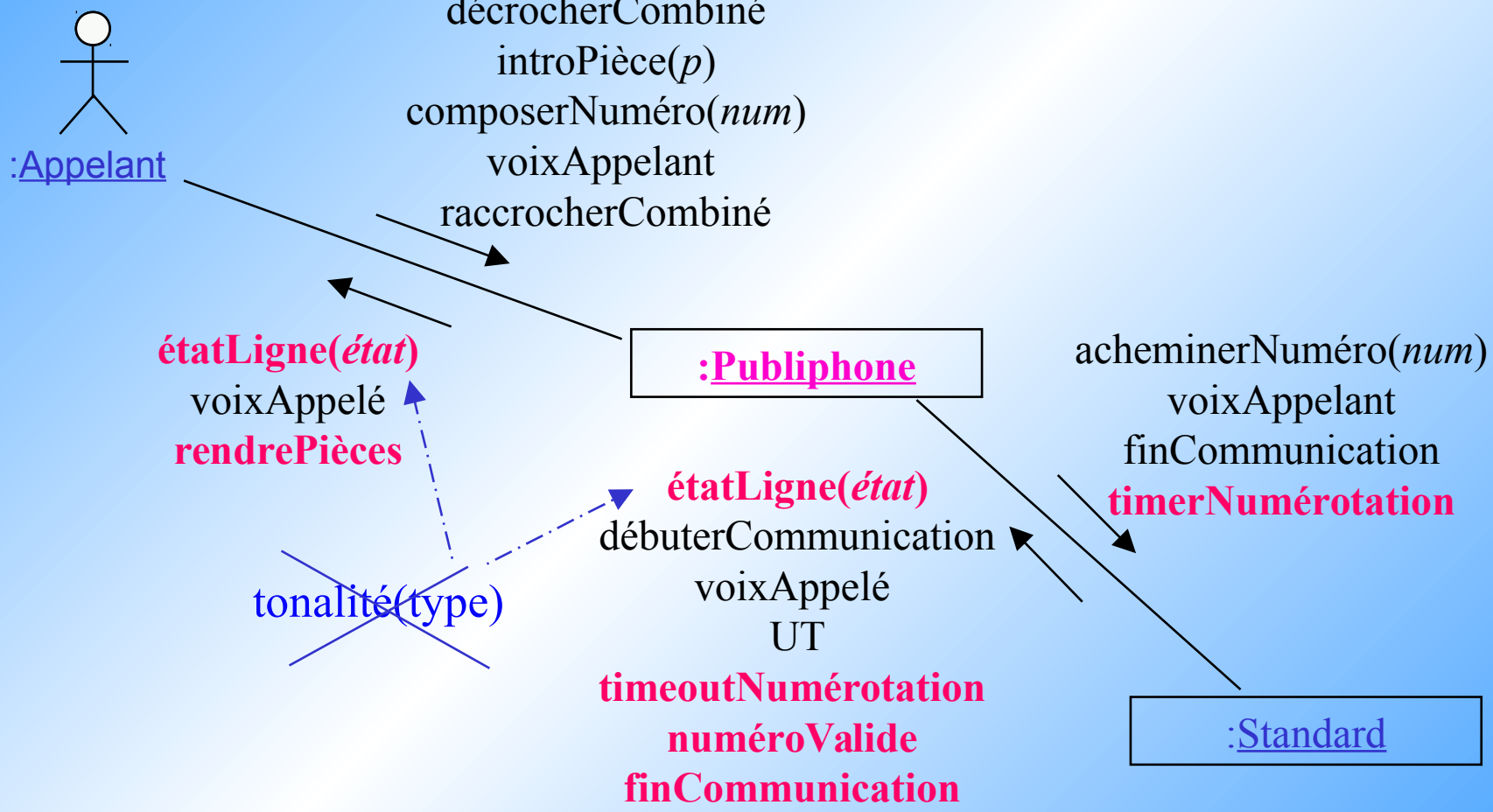
- si l'appelé raccroche le premier, la fin de communication est signalée par le standard (*finCommunication*) ;

- plus généralement, le standard transmet l'état de la ligne au Publiphone (*libre, occupée, en dérangement, etc.*) et pas seulement le type de tonalité (*étatLigne(état)*) .

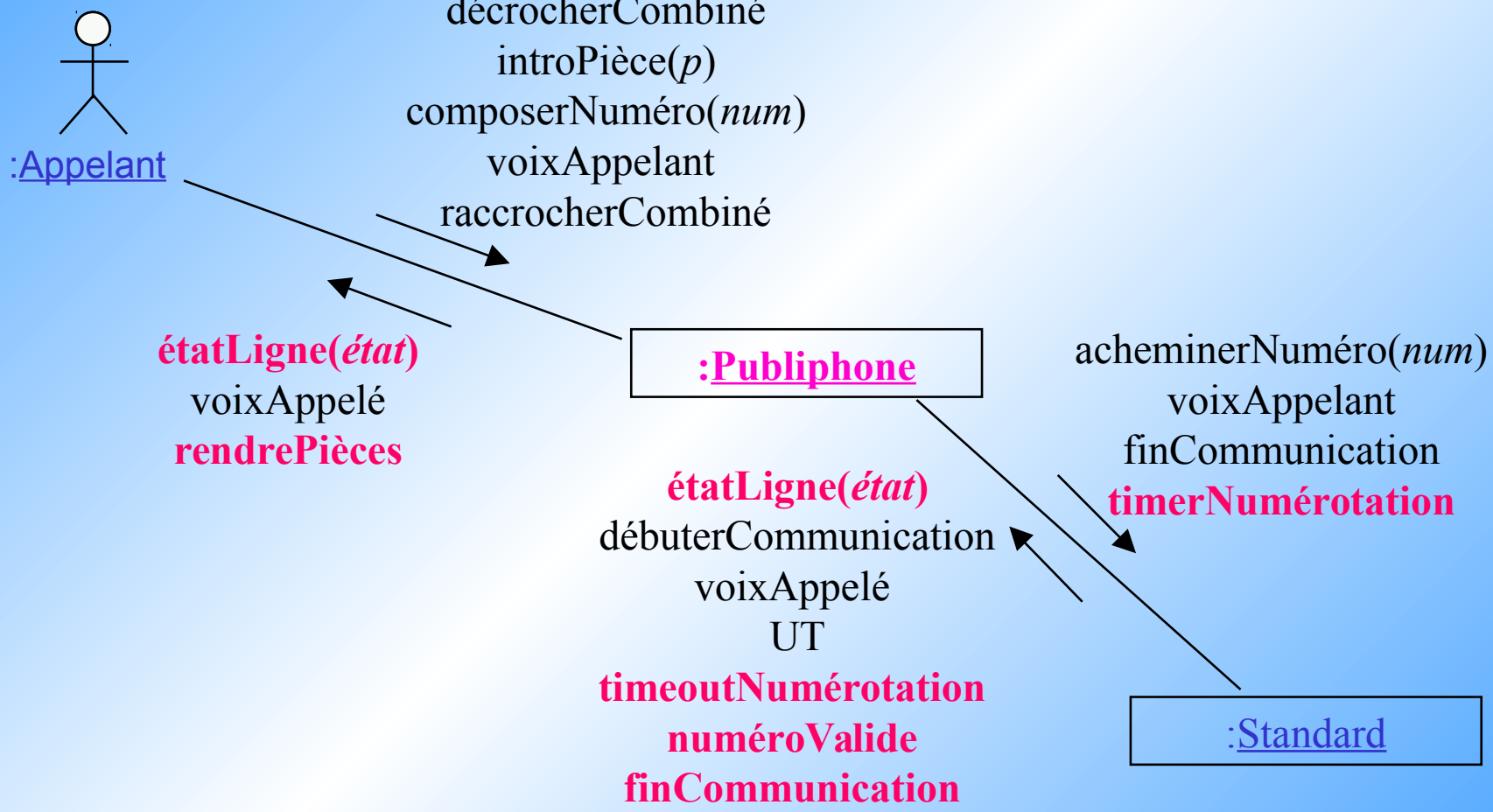
ÉTUDE DE CAS

Compléter le diagramme de contexte dynamique du Publiphone

SOLUTION


décrocherCombiné
introPièce(*p*)
composerNuméro(*num*)
voixAppelant
raccrocherCombiné
:Appelant
étatLigne(*état*)
voixAppelé
rendrePièces
:Publiphone
acheminerNuméro(*num*)
voixAppelant
finCommunication
timerNumérotation
étatLigne(*état*)
débuterCommunication
voixAppelé
UT
timeoutNumérotation
numéroValide
finCommunication
tonalité(type)
:Standard

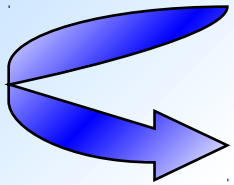
SOLUTION


:Appelant
décrocherCombiné
introPièce(p)
composerNuméro(num)
voixAppelant
raccrocherCombiné
étatLigne(état)
voixAppelé
rendrePièces
:Publiphone
acheminerNuméro(num)
voixAppelant
finCommunication
timerNumérotation
étatLigne(état)
débuterCommunication
voixAppelé
UT
timeoutNumérotation
numéroValide
finCommunication
:Standard

ÉTUDE DE CAS

- Après ce travail préliminaire, on va entreprendre une description exhaustive de la dynamique du Publiphone.

- Le comportement du Publiphone n'est pas trivial (*comme en témoigne le nombre élevé de messages identifiés sur le diagramme de contexte dynamique*).



Une approche itérative et incrémentale est préconisée.

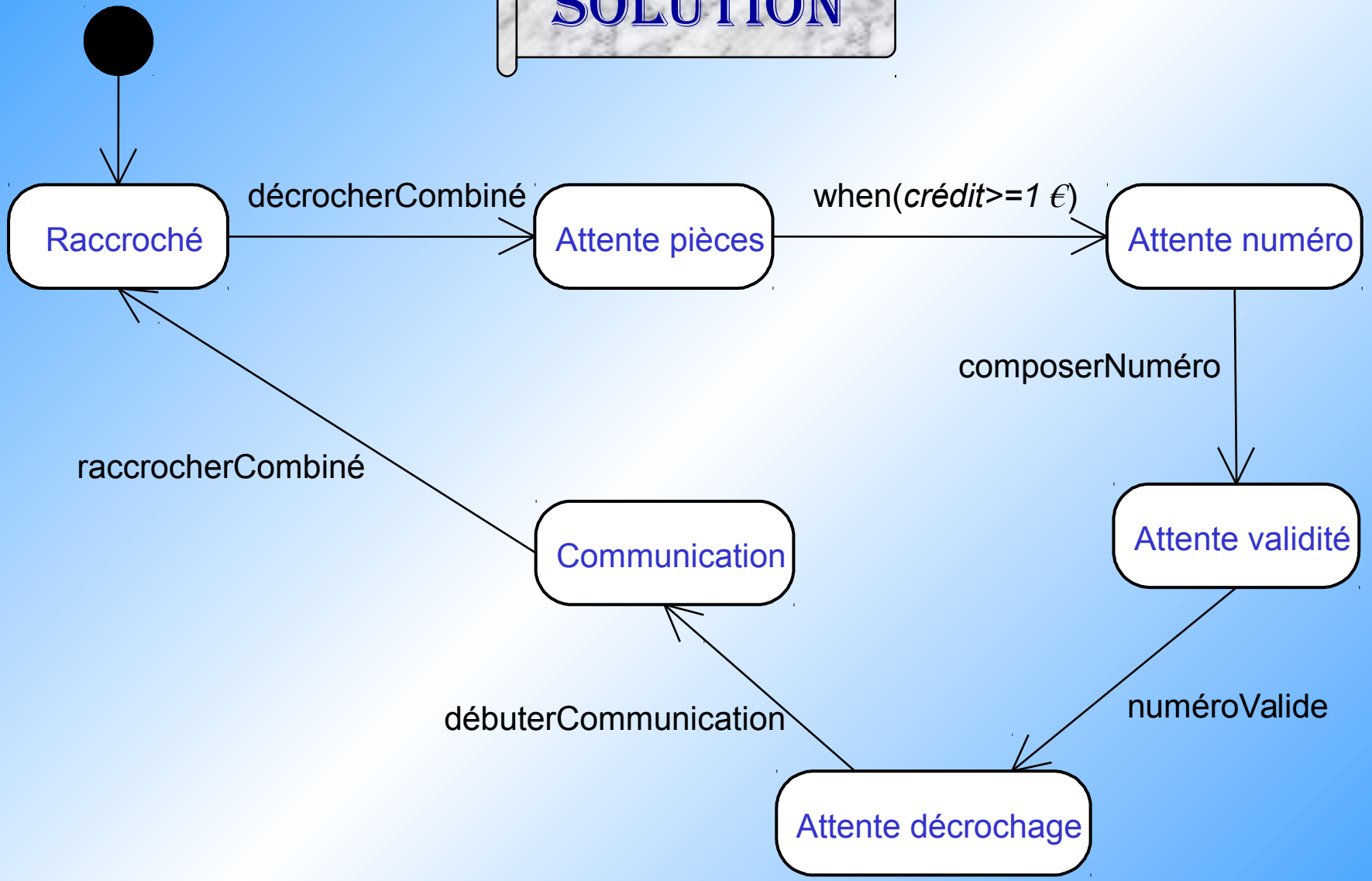
ÉTUDE DE CAS

Réaliser le diagramme d'états qui décrive le comportement nominal du Publiphone d'après le diagramme de séquence système.

SOLUTION

Prendre une feuille
mode paysage
crayon à papier

SOLUTION


Raccroché
décrocherCombiné
Attente pièces
when(*crédit*>=1 €)
Attente numéro
composerNuméro
Attente validité
numéroValide
Attente décrochage
débuterCommunication
Communication
raccrocherCombiné

SOLUTION

- L'état initial nominal du Publiphone est à *raccroché en service*.

- Quand l'appelant décroche le combiné, il doit ensuite introduire un minimum de *1* € pour pouvoir composer son numéro.

- Une fois qu'un numéro valide est composé, le Publiphone attend la réponse du standard, puis que l'appelé décroche.

- La conversation est alors établie jusqu'à ce que l'un des deux raccroche.

- Le Publiphone revient alors à son état initial.

SOLUTION

- La plupart des événements qui déclenchent les transitions entre états correspondent à la réception d'un message émis par un acteur : utilisation des mêmes noms pour les événements que pour les messages correspondants (*sauf pour numeroValide qui est plus lisible que validiteNumero(vrai)*).

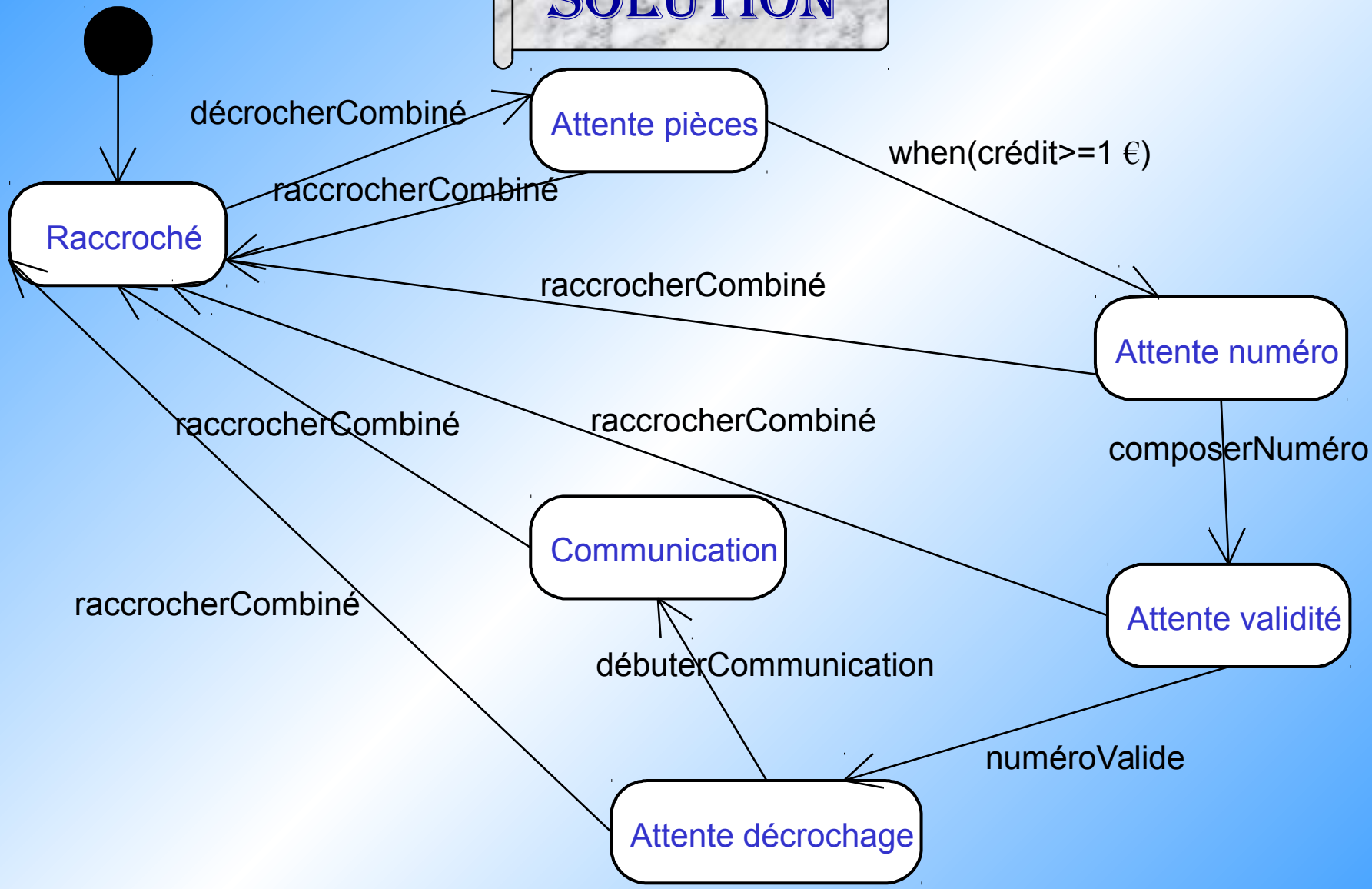
- Seul le passage de l'état "*Attente pièces*" à l'état "*Attente numéro*" est provoqué par un événement interne au Publiphone : la détection du dépassement du seuil de *1* €.

- UML propose un mot-clé pour distinguer ces changements d'états internes : ***when***, suivi d'une expression booléenne dont le passage de faux à vrai déclenche la transition.

ÉTUDE DE CAS

Représenter le fait que l'appelant peut raccrocher à tout moment et pas seulement dans l'état conversation ?

SOLUTION


décrocherCombiné
Attente pièces
when(crédit>=1 €)
raccrocherCombiné
Raccroché
raccrocherCombiné
Attente numéro
raccrocherCombiné
raccrocherCombiné
composerNuméro
Communication
raccrocherCombiné
Attente validité
débuterCommunication
numéroValide
Attente décrochage

SOLUTION

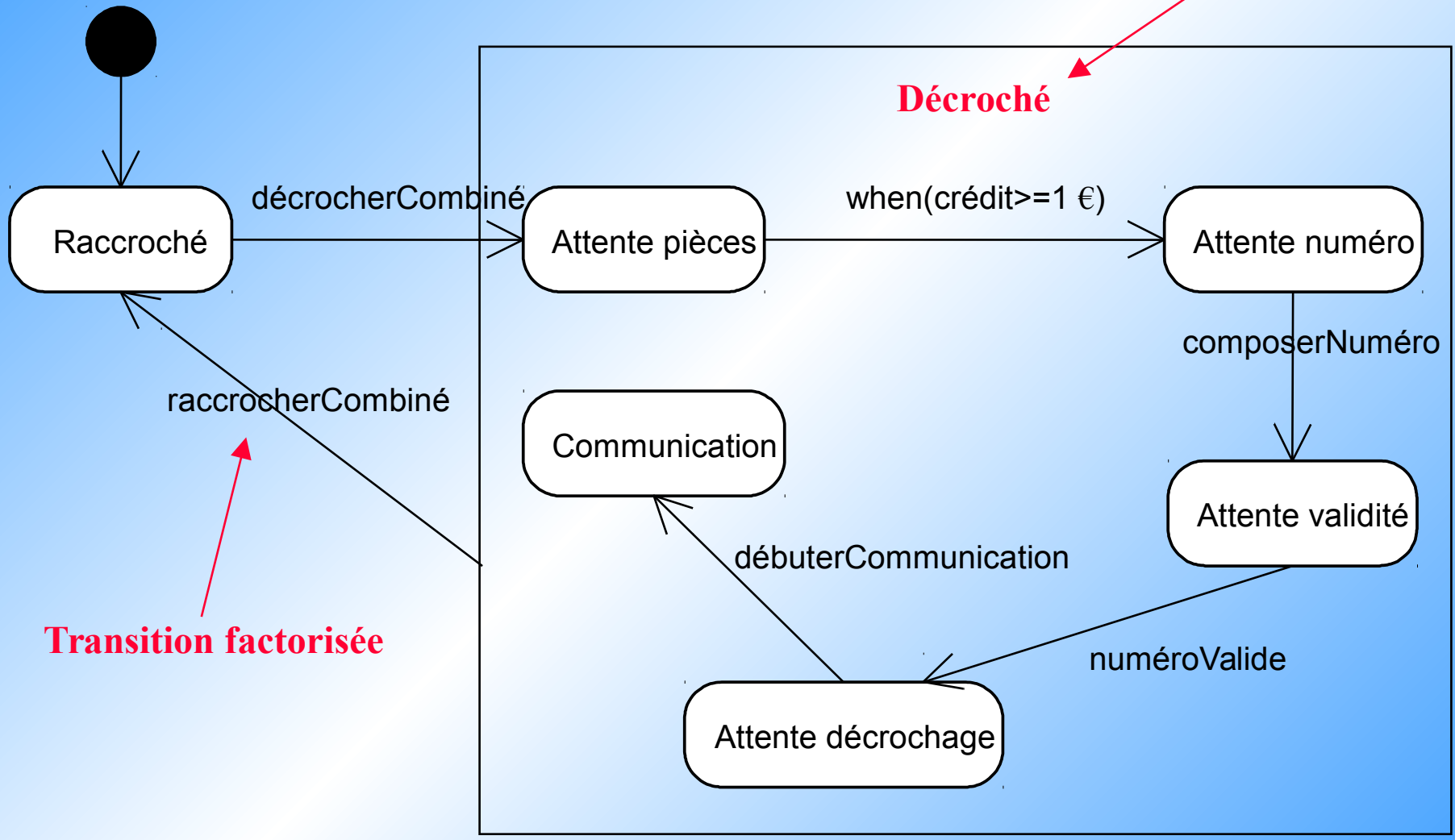
La solution triviale consiste à ajouter des transitions déclenchées par l'événement *raccrocherCombiné* et sortant de tous les états pour amener vers l'état *Raccroché*.

Mais le diagramme paraît soudain bien chargé …

SOLUTION

La solution élégante consiste à introduire un super-état, *Décroché*, ce qui permet de factoriser la transition de sortie vers l'état *Raccroché*.

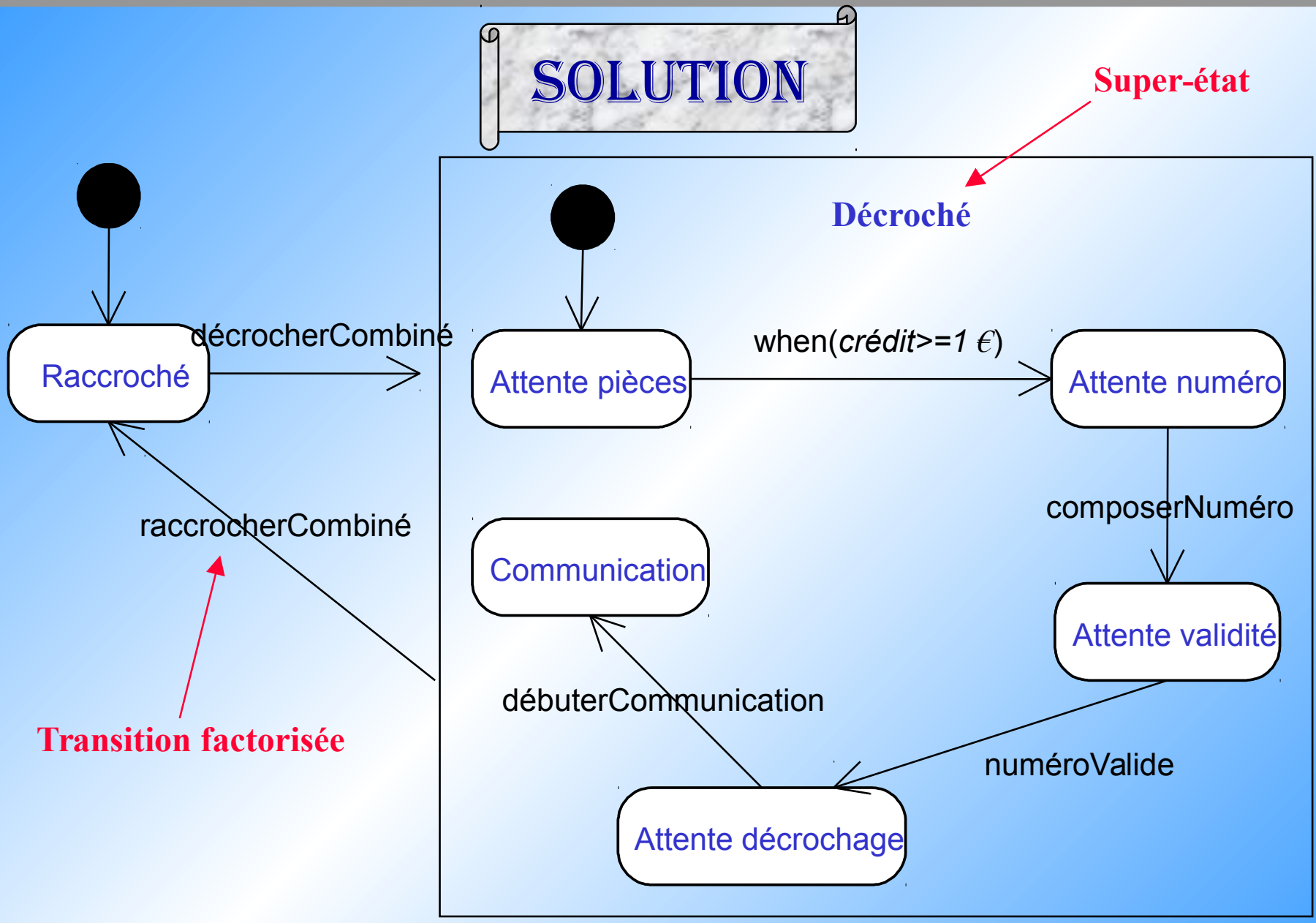
SOLUTION


Super-état
Décroché
Raccroché
décrocherCombiné
Attente pièces
when(crédit>=1 €)
Attente numéro
composerNuméro
raccrocherCombiné
Communication
Attente validité
débuterCommunication
numéroValide
Attente décrochage
Transition factorisée

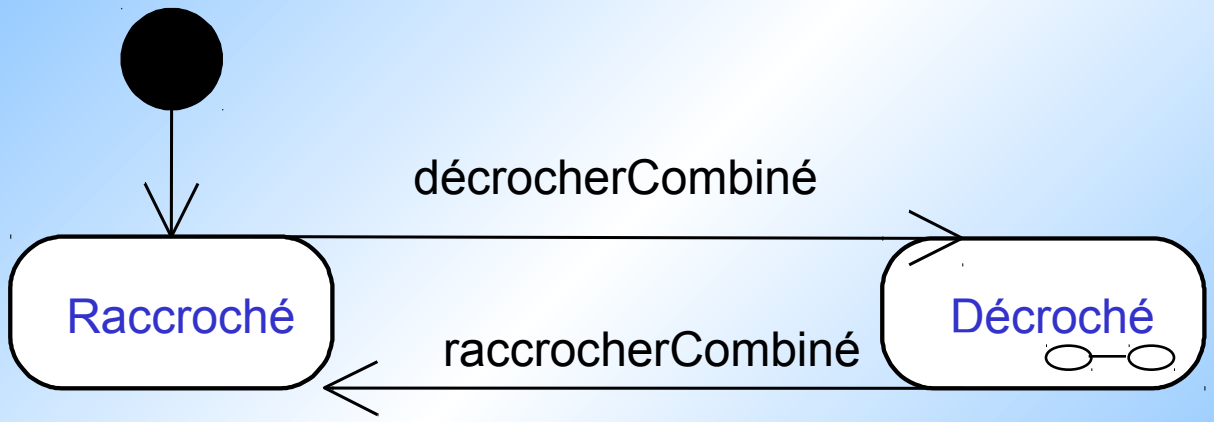
SOLUTION

On peut utiliser une notation plus sophistiquée pour la transition de *Raccroché* vers *Attente pièces*.

SOLUTION


Super-état
Décroché
Raccroché
décrocherCombiné
Attente pièces
when(crédit>=1 €)
Attente numéro
composerNuméro
raccrocherCombiné
Communication
Attente validité
débuterCommunication
numéroValide
Attente décrochage
Transition factorisée

SOLUTION



super - état

SOLUTION

Cette manière de procéder permet de découper le diagramme d'états en deux niveaux :

- Un premier niveau ne faisant apparaître que les états *Raccroché* et *Décroché* ;

- Un second niveau correspondant à la décomposition de *Décroché*.

ÉTUDE DE CAS

Comment le crédit de l'appelant peut-il atteindre *1 €* ?

Prendre en compte le message *introPièce(p)*

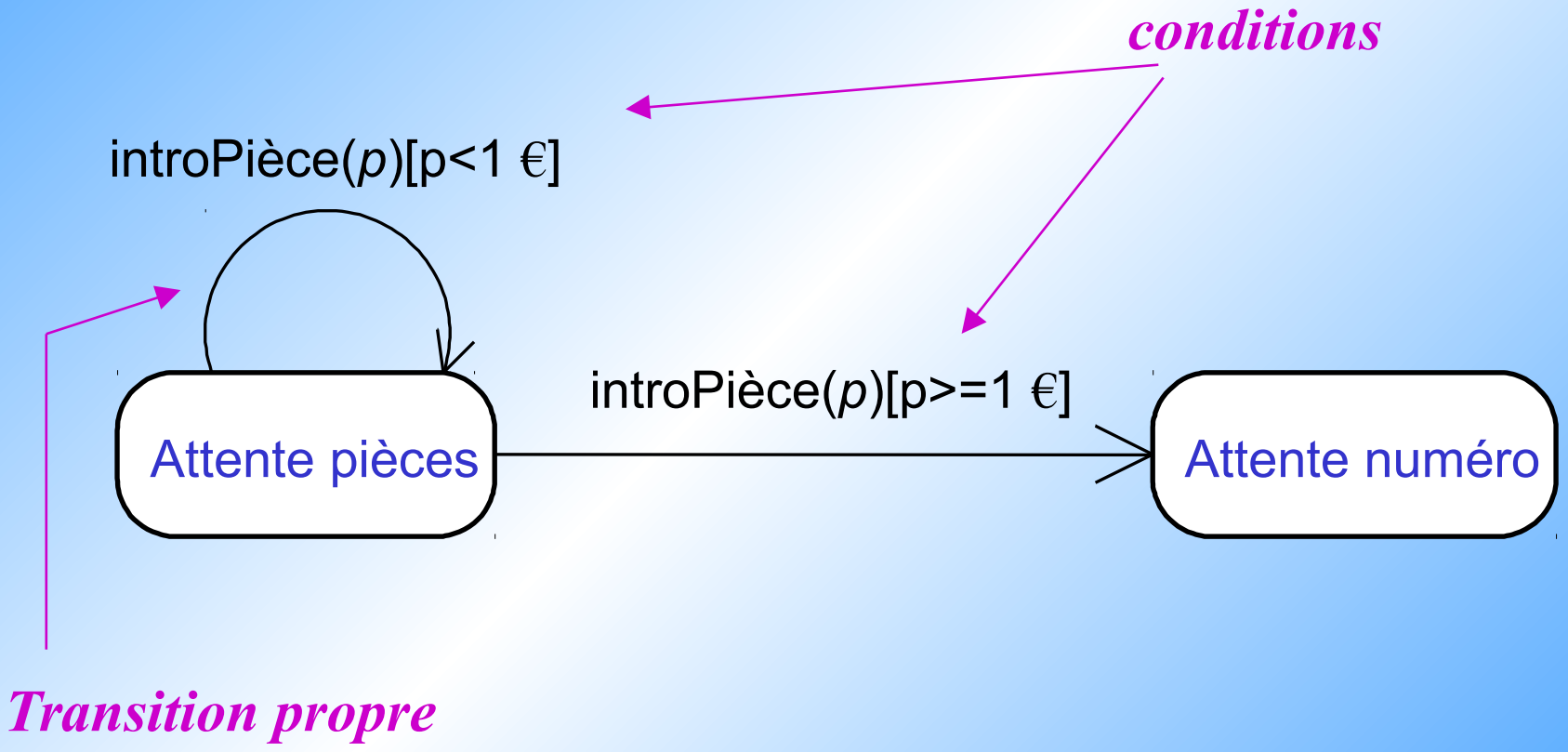
SOLUTION

Pour le moment, le crédit de l'appelant n'intervient que dans l'expression booléenne associée à l'événement interne *when*.
Cependant, pour que le crédit atteigne *1* €, il faut que l'appelant introduise une ou plusieurs pièces.

On peut donc placer une transition propre (*qui ramène vers l'état de départ*) sur l'état *Attente pièces*.
Dès que le crédit dépasse *1* €, le Publiphone doit passer dans l'état *Attente numéro*.
Si on ne souhaite utiliser que des événements de réception de messages, il faut introduire des conditions sur les transitions.

SOLUTION

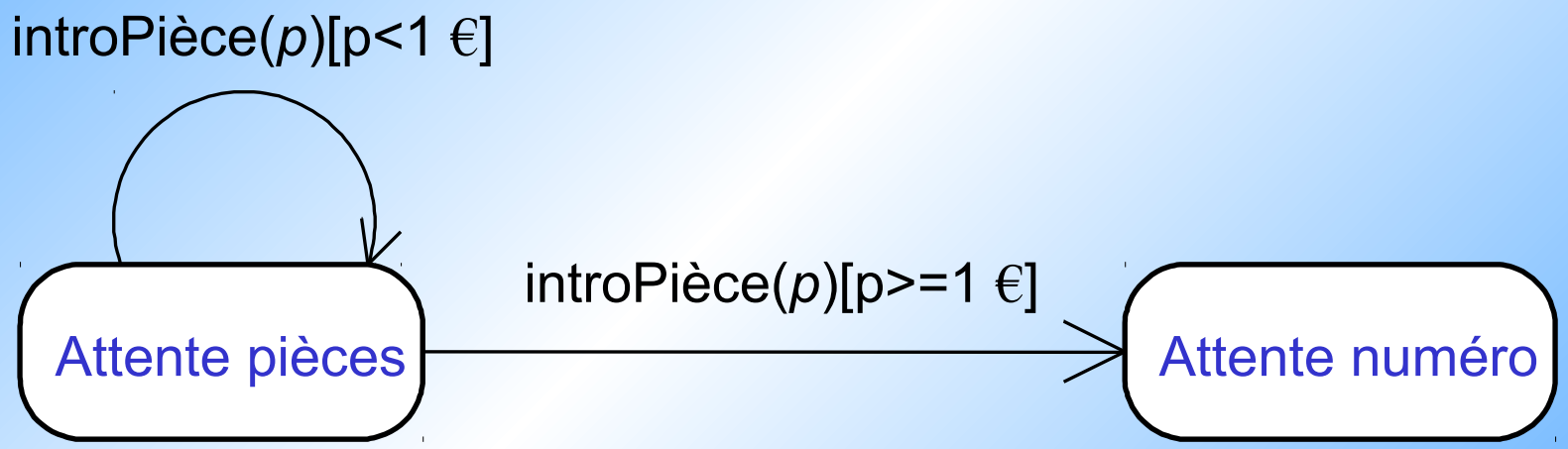
Que pensez-vous de cette solution ?



SOLUTION

Que pensez-vous de cette solution ?

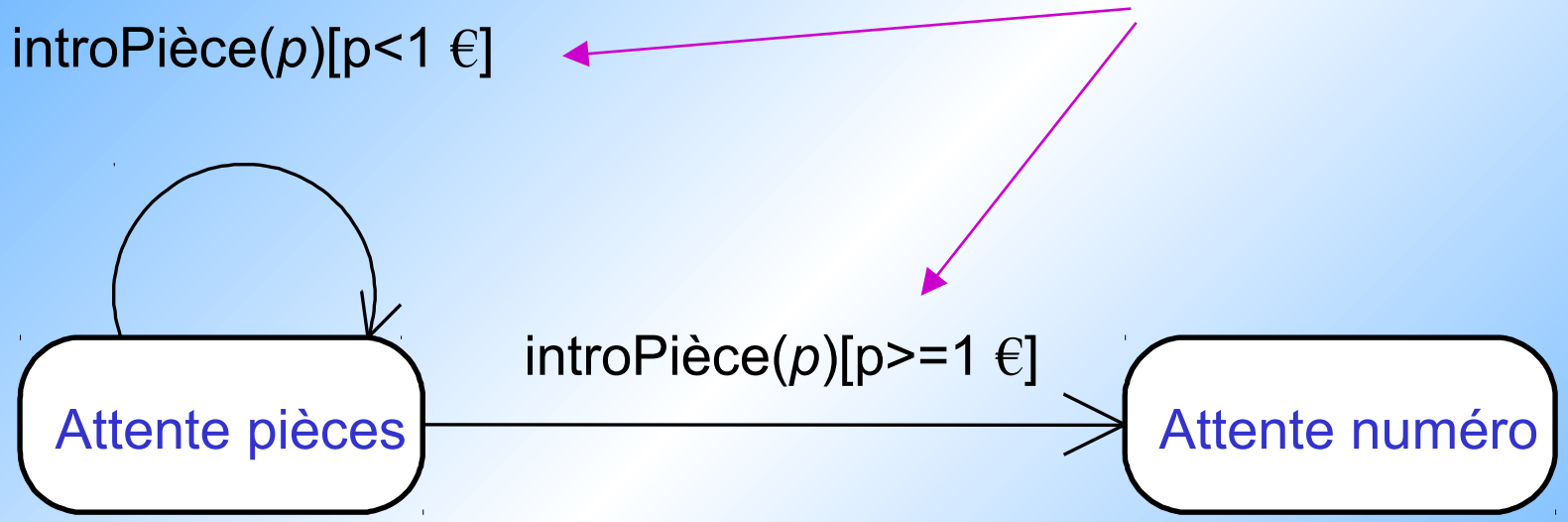
conditions



Transition propre

SOLUTION

Que pensez-vous de cette solution ?



Passage de l'état Attente pièces vers Attente numéro :
si introduction d'une pièce de 1 ou 2 €

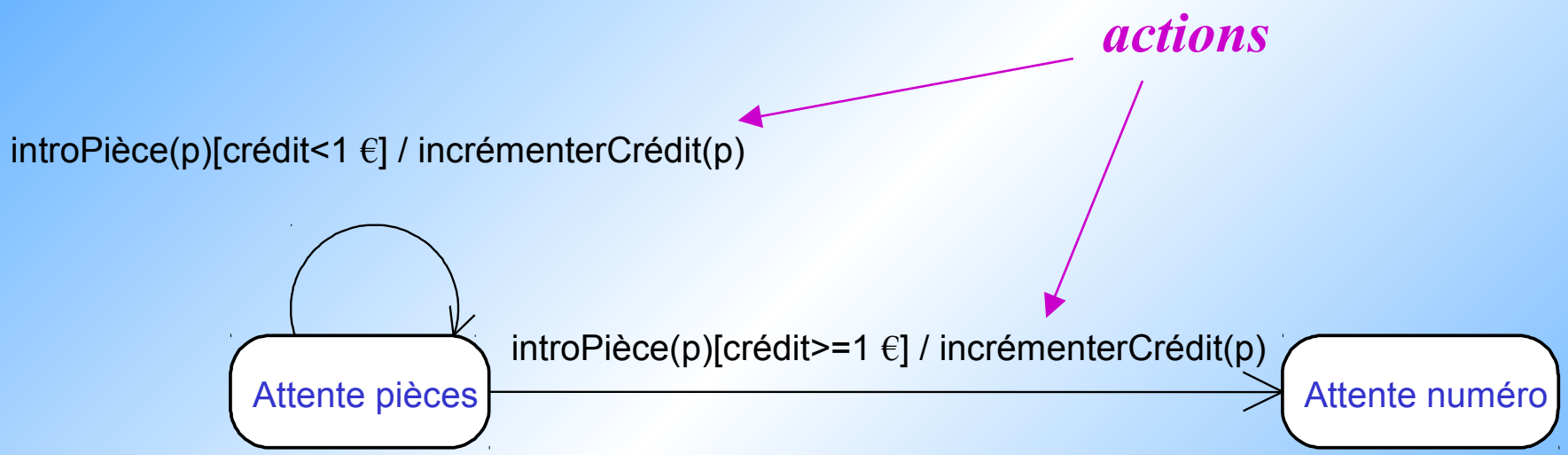
variable qui comptabilise : crédit

SOLUTION

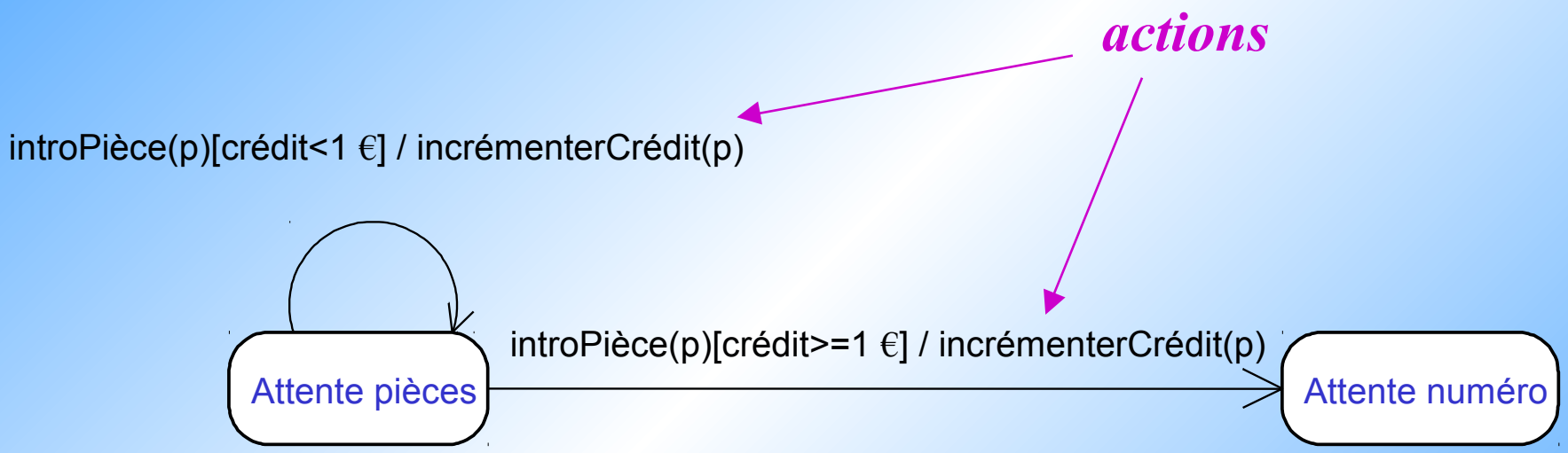
- Cette solution est erronée puisqu'elle interdit de composer un numéro après avoir introduit plusieurs pièces totalisant *1* €.

- Il faut donc que le Publiphone mémorise un attribut *crédit* qui est incrémenté à chaque introduction de pièce.

SOLUTION


actions
introPièce(p)[crédit<1 €] / incrémenterCrédit(p)
introPièce(p)[crédit>=1 €] / incrémenterCrédit(p)
Attente pièces
Attente numéro

SOLUTION



~~Introduction d'une pièce~~
~~test~~
~~incrémentation~~

Introduction d'une pièce
incrémentation
test

SOLUTION

Solution est également fausse (*mais d'une manière plus subtile* !)

En effet, la sémantique d'une transition en UML est la suivante : lorsque l'événement déclencheur se produit, la condition est testée. Si la condition est évaluée à vrai, la transition est déclenchée et l'action associée est alors réalisée.

Voyons comment cela se passe concrètement sur notre exemple, en partant de *Attente pièces* avec un crédit initial de *0* €.

- l'appelant introduit une pièce de *0,50* €. Le crédit est inférieur à *1* € (*il vaut 0* €). La transition propre est déclenchée et le crédit vaut maintenant *0,50* €.

SOLUTION

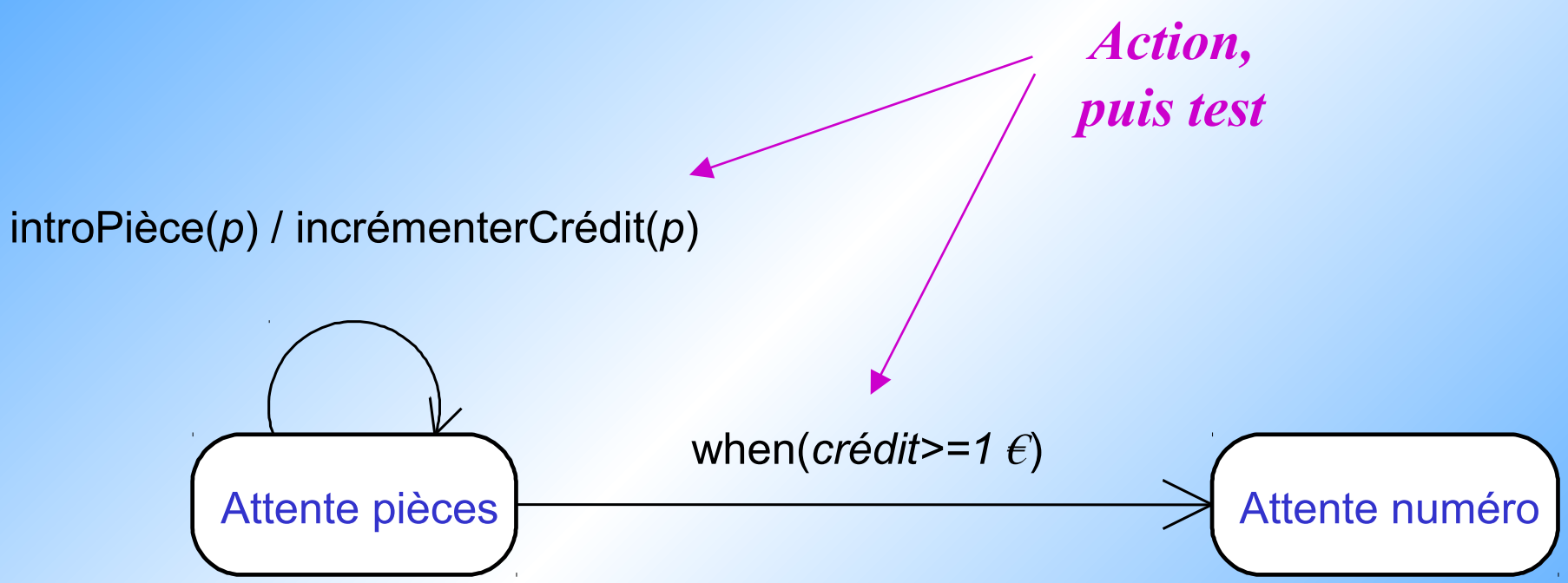
- l'appelant introduit une pièce de *0,50* €. Le crédit est inférieur à *1* € (*il vaut 0,50* €). La transition propre est déclenchée et le crédit vaut maintenant *1* €.

SOLUTION

Le résultat est surprenant : l'appelant a payé *1* € et ne peut toujours pas composer son numéro…

La moralité de ce constat est la suivante : n'essayez pas de tester une donnée dans une condition avant de l'avoir modifiée par une action, ce qui est le cas lorsqu'on veut tout faire tenir dans une seule transition.

SOLUTION



Introduction pièce
Incrémentation

Test

ÉTUDE DE CAS

Compléter la gestion du crédit de l'appelant

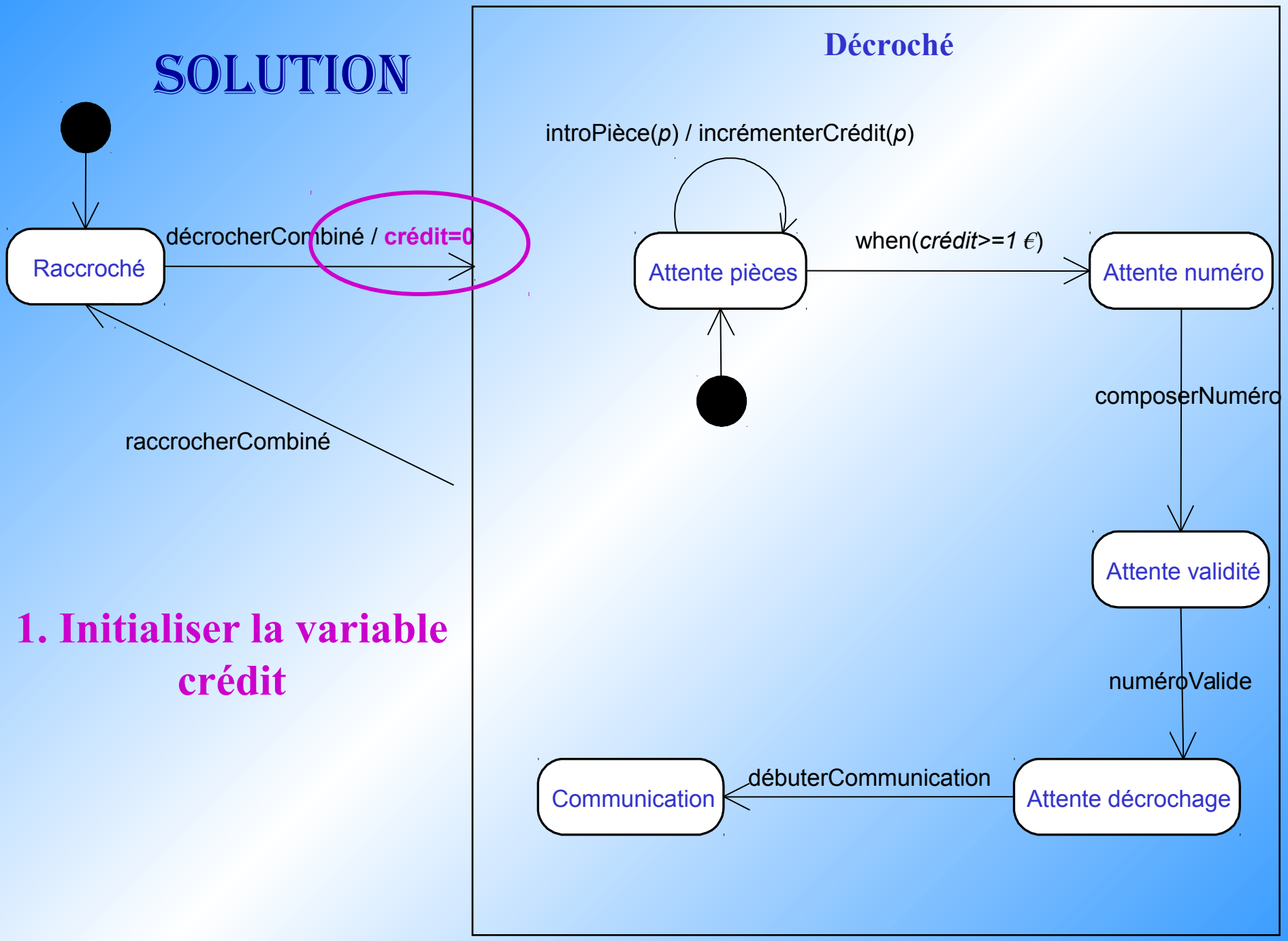
- Initialiser la variable crédit
- Décrémenter le crédit ou taxer
- Rendre les pièces

ÉTUDE DE CAS

Compléter la gestion du crédit de l'appelant

- Initialiser la variable crédit

SOLUTION



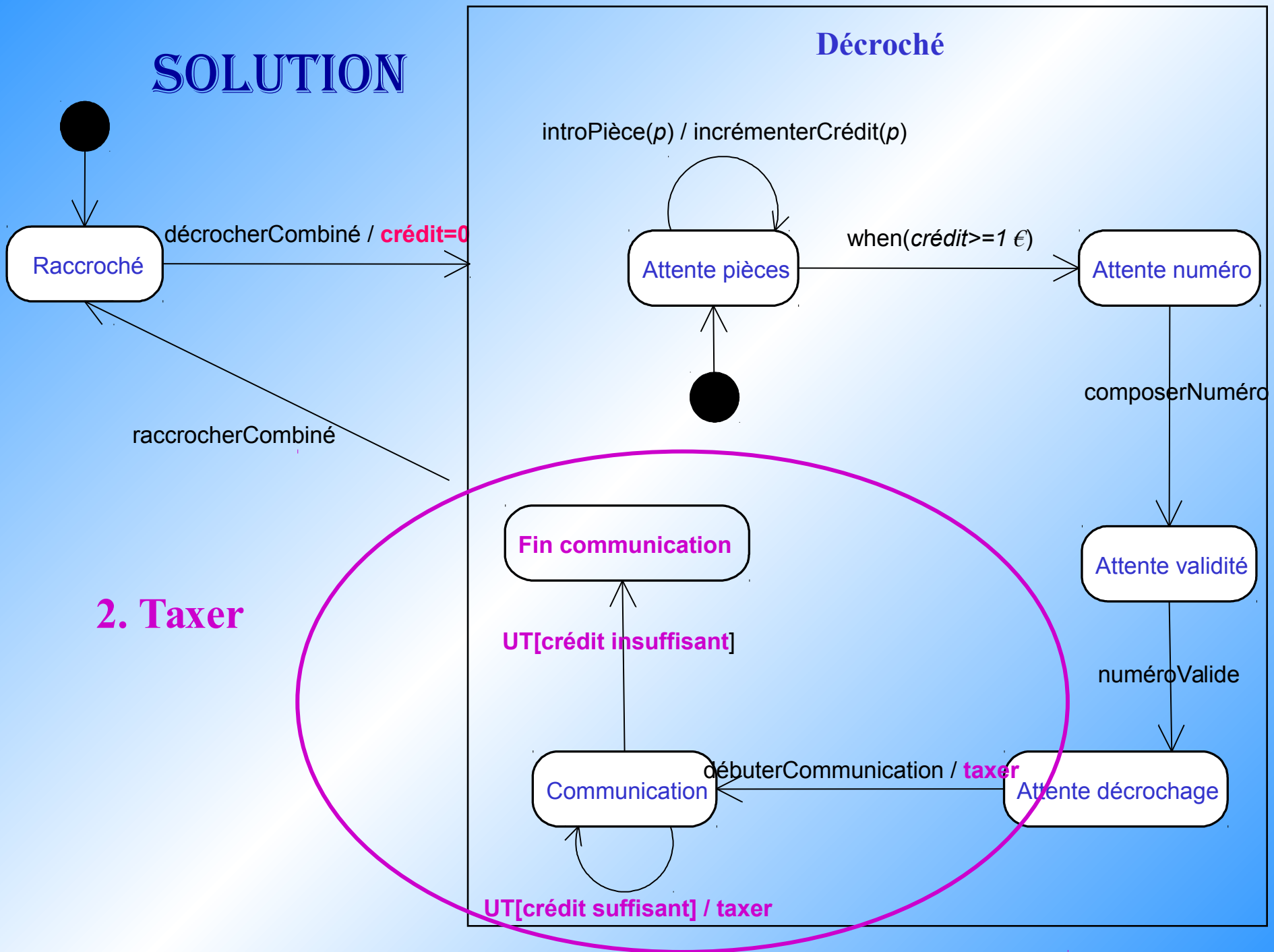
1. Initialiser la variable crédit

ÉTUDE DE CAS

Compléter la gestion du crédit de l'appelant

- Décrémenter le crédit ou taxer

SOLUTION

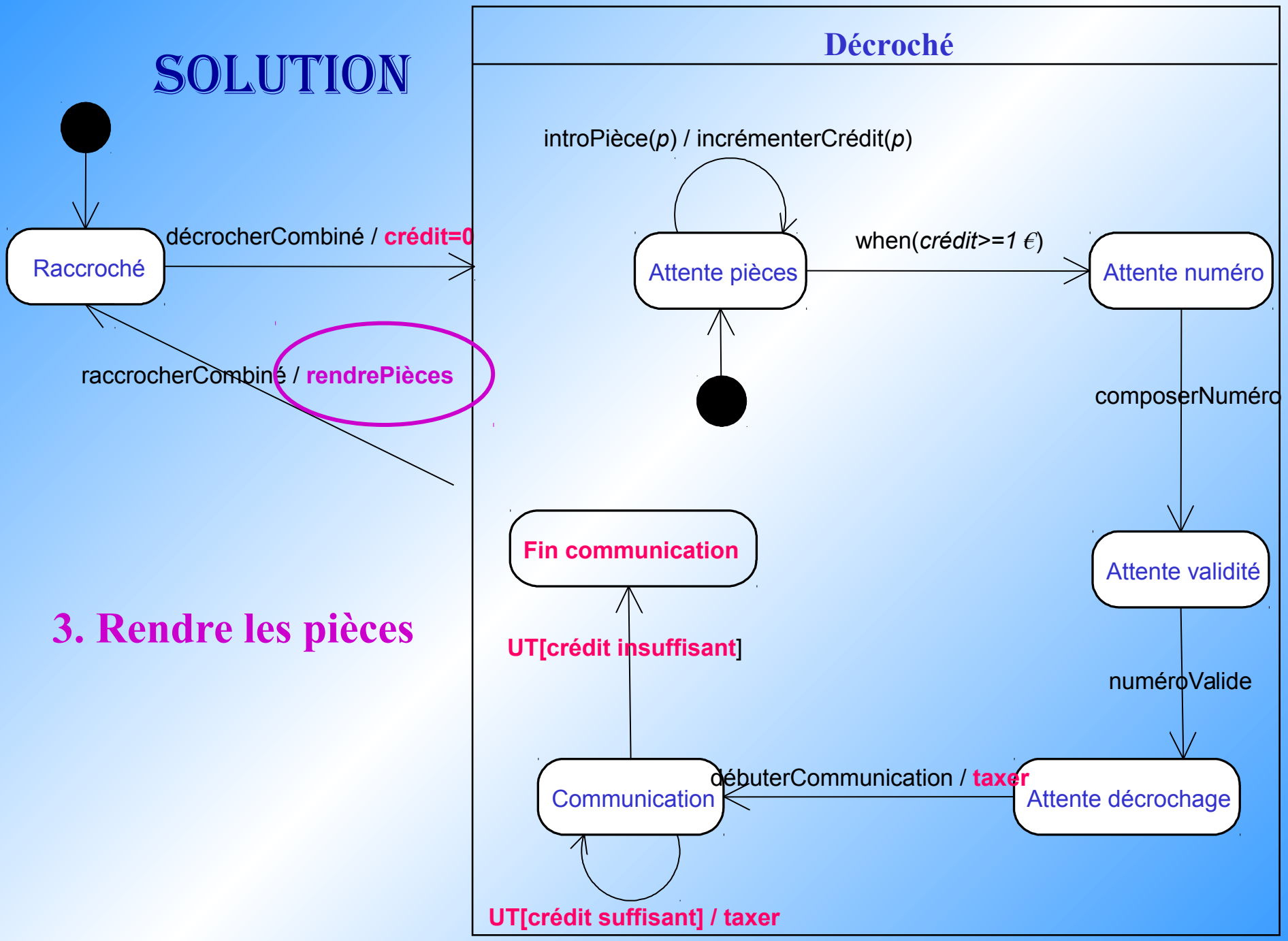

Décroché
introPièce(p) / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
Attente pièces
when(crédit>=1 €)
Attente numéro
composerNuméro
raccrocherCombiné
Fin communication
Attente validité
2. Taxer
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer

ÉTUDE DE CAS

Compléter la gestion du crédit de l'appelant

- Rendre les pièces

SOLUTION



3. Rendre les pièces

Étude de cas

Modéliser le fait que l'appelant peut introduire des pièces à tout moment

SOLUTION

• Admettons que le fait de décrocher le combiné fasse tomber les éventuelles pièces qui auraient été introduites auparavant. Le crédit doit donc être initialisé à *0* sur cette transition (*décrocherCombiné / crédit = 0*).

• Ensuite, pour que le crédit atteigne *1* €, il faut que l'appelant introduise une ou plusieurs pièces. On a donc placé une transition propre sur l'état *Attente pièces* (*vu précédemment*).

• Dès que le crédit dépasse *1* €, la transition de changement *when* amène le Publiphone en l'état *Attente numéro* (*when(crédit >= 1€)*).

SOLUTION

• Le crédit n'évolue plus jusqu'à ce que l'appelé décroche, ce qui se traduit par le message *débuterCommunication* émis par le standard.
En effet, à partir de ce moment là, le crédit est décrémenté régulièrement comme l'indique la phrase *5* (*le Publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps générée par le standard*). L'action *taxer* représente la chute d'une pièce à chaque impulsion.

• Enfin, il ne faut pas oublier de rendre les pièces non utilisées quand l'appelant raccroche (*raccrocherCombiné / rendrePièces*).

SOLUTION

• On notera l'introduction du nouvel état *Fin communication* pour parer au cas où la communication est interrompue par le Publiphone faute de crédit suffisant (*UT[crédit insuffisant]*).

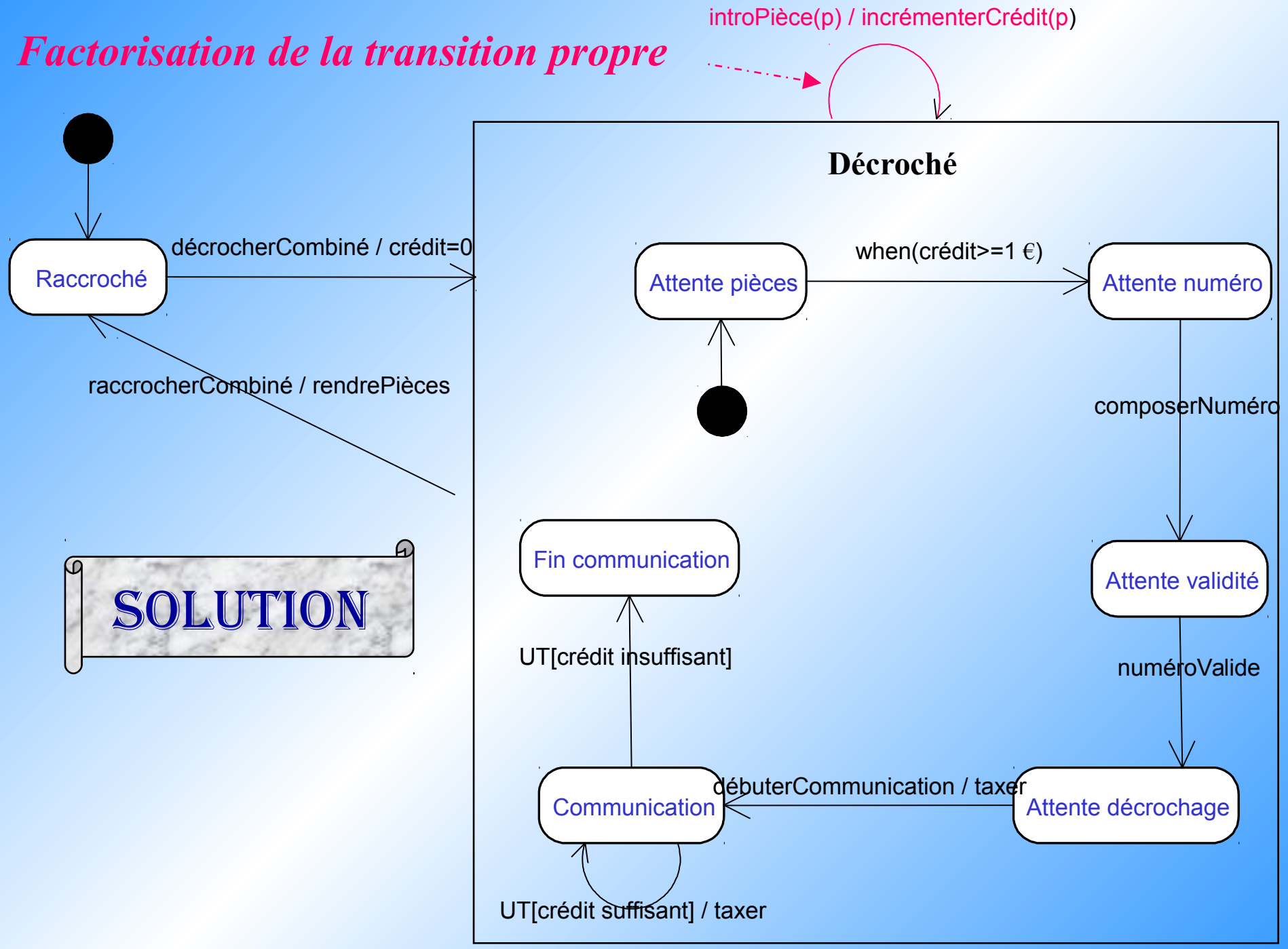
SOLUTION

Il nous reste encore à modéliser la
Phrase 6 : *on peut ajouter des pièces à tout moment*.

Là encore, plusieurs solutions sont possibles.

La première idée consiste à introduire une transition propre identique à celle de *Attente pièces* sur chaque sous-état de *Décroché*.
Cette façon de procéder est correcte, mais le dispositif est très lourd, et on va chercher à factoriser au moyen du super-état.

Factorisation de la transition propre



SOLUTION

Transition propre ou transition interne

Dans le cas d'une **transition propre**, l'objet quitte son état de départ pour le retrouver. Cela peut avoir des effets secondaires non négligeables comme l'interruption puis le redémarrage d'une activité, la réalisation d'actions en entrée (*entry*) ou en sortie (*exit*) de l'état, etc.
De plus, quand un état est décomposé en sous-états, une transition propre ramène forcément l'objet dans le sous-état initial.

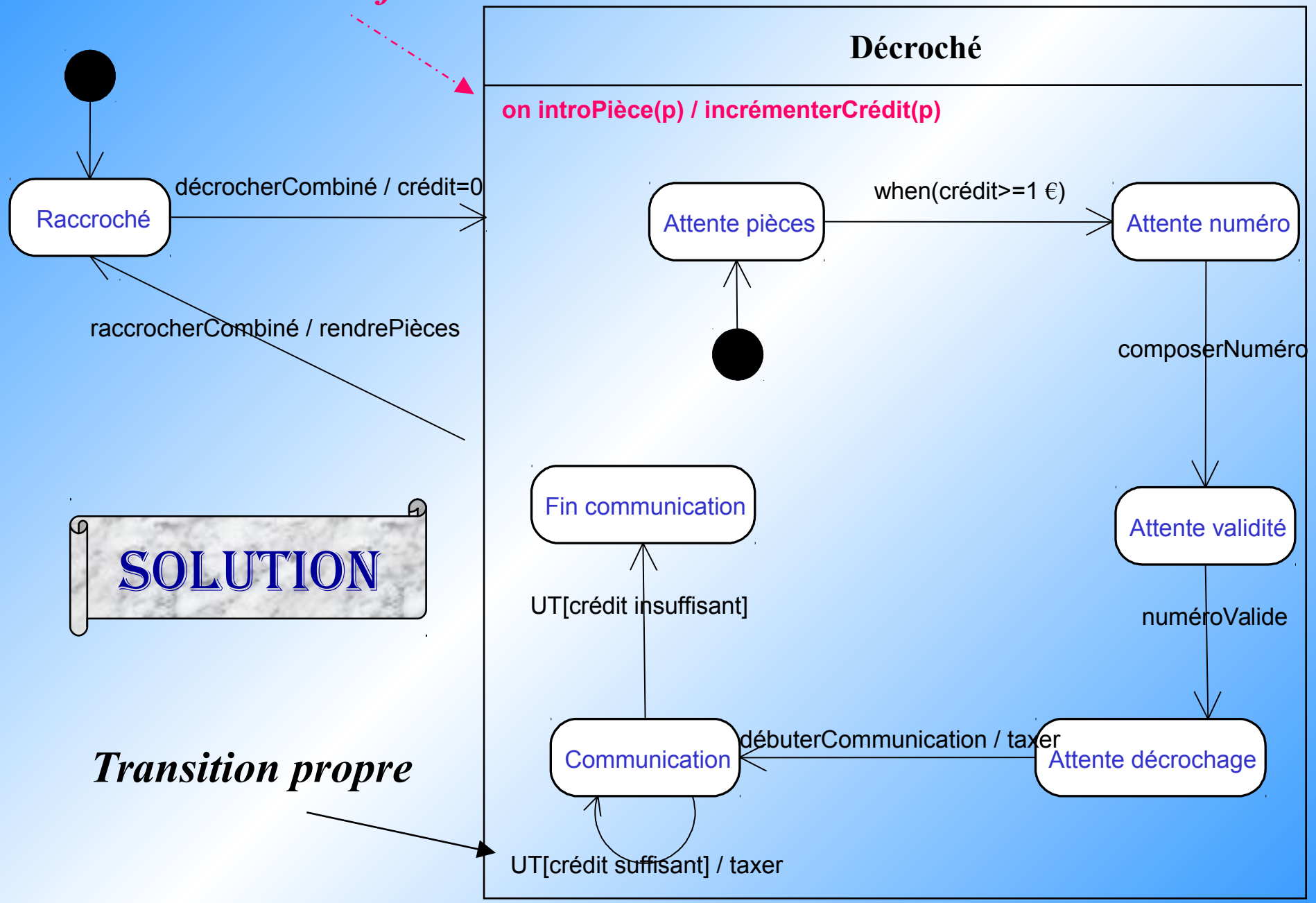
Ici, chaque introduction de pièce ramènerait le Publiphone dans l'état *Attente pièce*, qui est implicitement sous-état initial de *Décroché* !

SOLUTION

Transition propre ou transition interne

Pour résoudre ce problème courant, il existe en UML la notion de **transition interne**. Elle représente un couple (*événement/action*) qui n'a aucune influence sur l'état courant.
La transition interne est ainsi notée à l'intérieur du symbole de l'état.

Transition interne factorisée



SOLUTION

Transition propre

SOLUTION

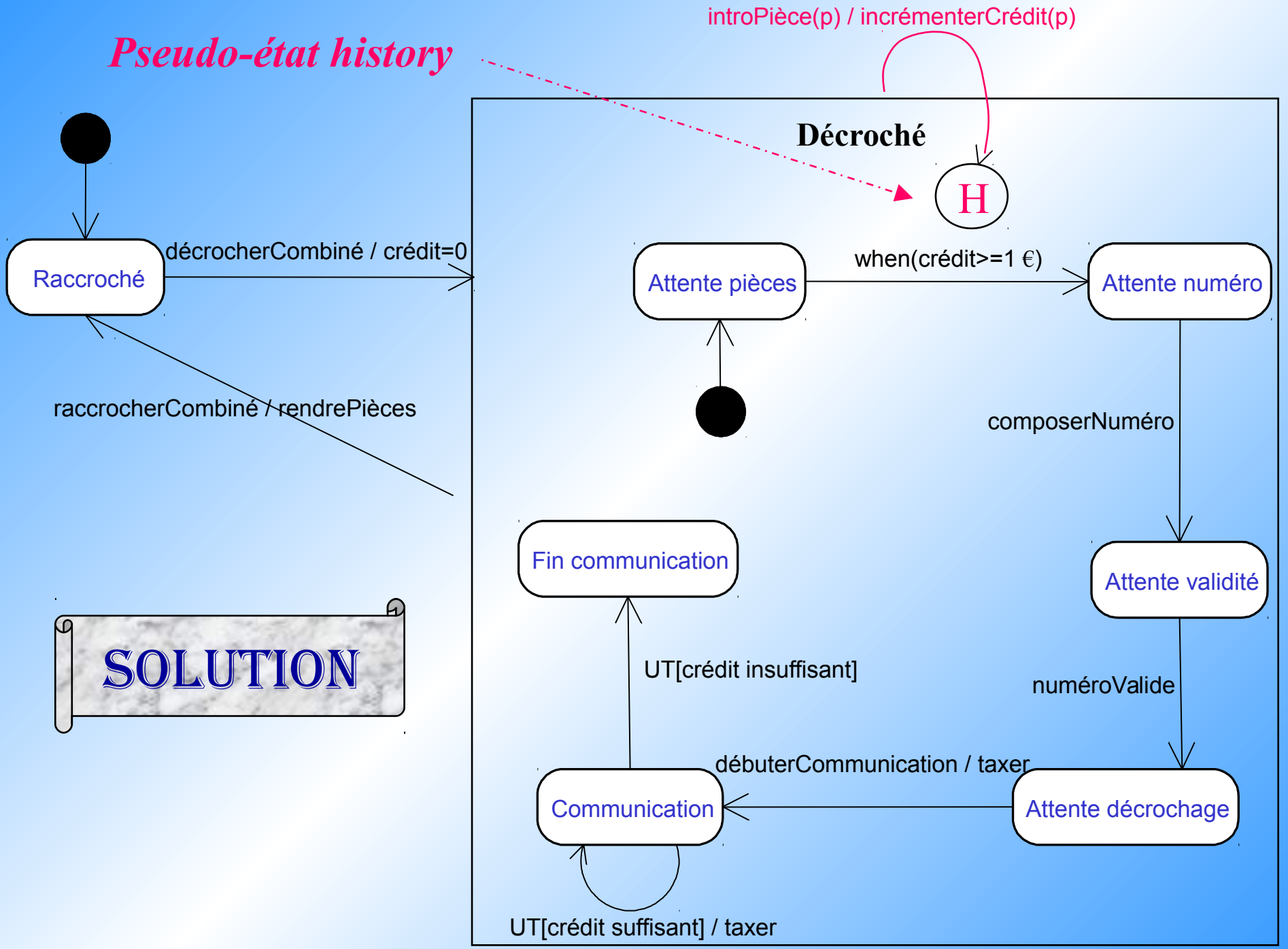
Une deuxième solution correcte, mais plus complexe, consiste à utiliser le pseudo-état *history*.

Pseudo-état history

L'activation du pseudo-état *history* permet à un super-état de se souvenir du dernier sous-état séquentiel qui était actif avant une transition sortante.

Une transition vers l'état *history* rend à nouveau actif le dernier sous-état actif, au lieu de ramener vers le sous-état initial.

Pseudo-état history



SOLUTION

ÉTUDE DE CAS

Compléter le diagramme d'états pour prendre en compte l'ensemble de l'énoncé

ÉTUDE DE CAS

Compléter le diagramme d'états pour prendre en compte l'ensemble de l'énoncé

Phrase 2 :

Après l'introduction de la monnaie, l'utilisateur a deux minutes pour composer son numéro
(*ce délai est décompté par le standard*).

ÉTUDE DE CAS

Compléter le diagramme d'états pour prendre en compte l'ensemble de l'énoncé

Phrase 2 :

Après l'introduction de la monnaie, l'utilisateur a deux minutes pour composer son numéro
(*ce délai est décompté par le standard*).

-timerNumérotation
-timeoutNumérotation

SOLUTION

Phrase 2 :
Après l'introduction de la monnaie, l'utilisateur a deux minutes pour composer son numéro (*ce délai est décompté par le standard*).

Le délai étant décompté par le standard, on a introduit deux messages dans le diagramme de contexte :
- *timerNumérotation* envoyé par le Publiphone au standard ;
- *timeoutNumérotation* envoyé par le standard au Publiphone.

SOLUTION

Envoi de message *send*

UML propose un mot-clé *send* pour représenter l'action importante qui consiste à envoyer un message à un autre objet sur déclenchement d'une transition.

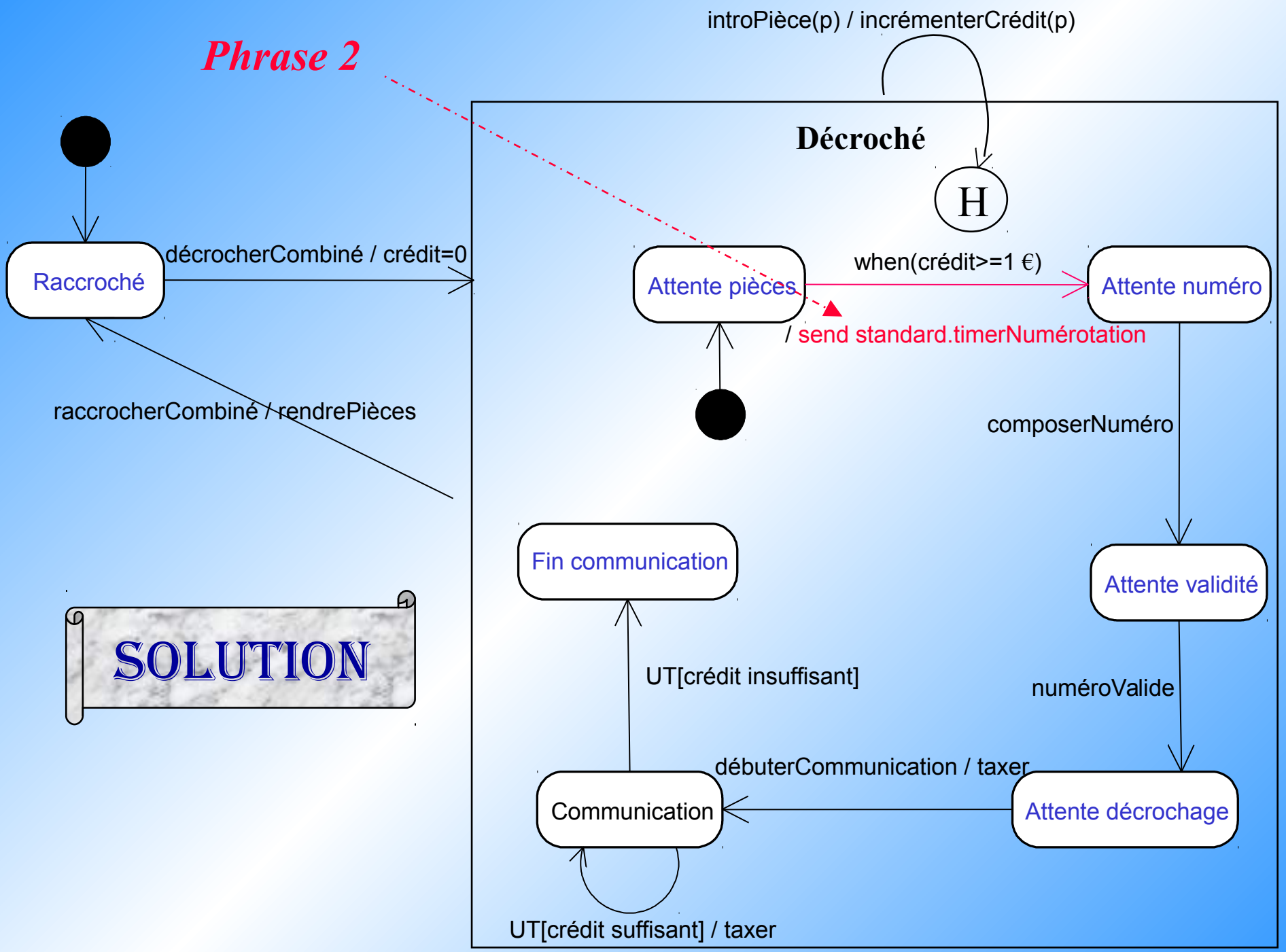
La syntaxe de cette action particulière est la suivante :
/ *send cible.message*

SOLUTION

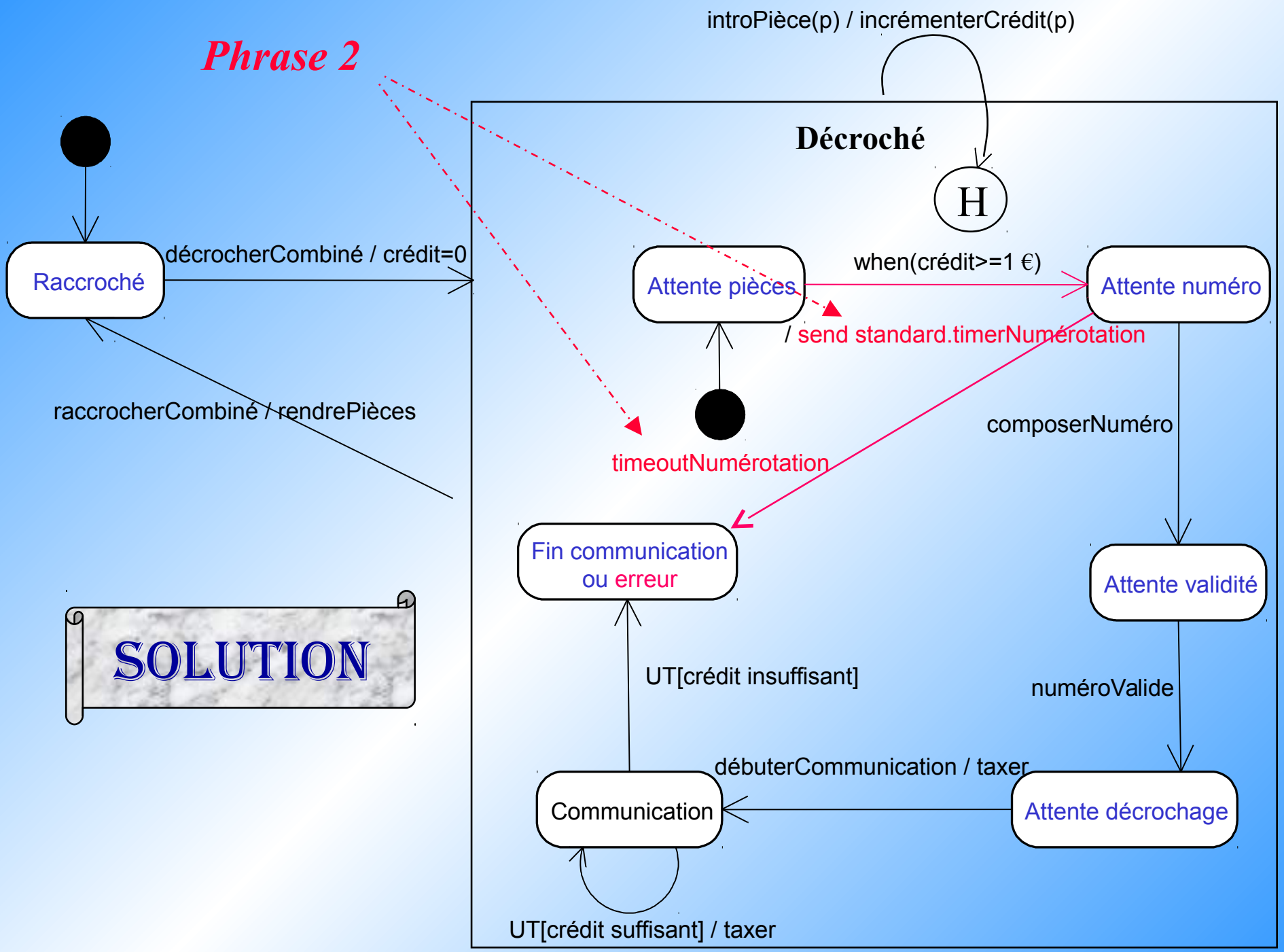
Dans le diagramme d'états du Publiphone, on a donc une transition qui sera déclenchée par la réception du message *timeoutNumérotation*, et l'envoi du message *timerNumérotation* à l'entrée de l'état *Attente numéro*.

Il faut noter que l'on a renommé l'état *Fin communication* en *Fin communication ou erreur* car cet *état puits* (*i.e. n'ayant aucune transition en sortie*) nous servira également pour tous les cas d'erreur.

Phrase 2

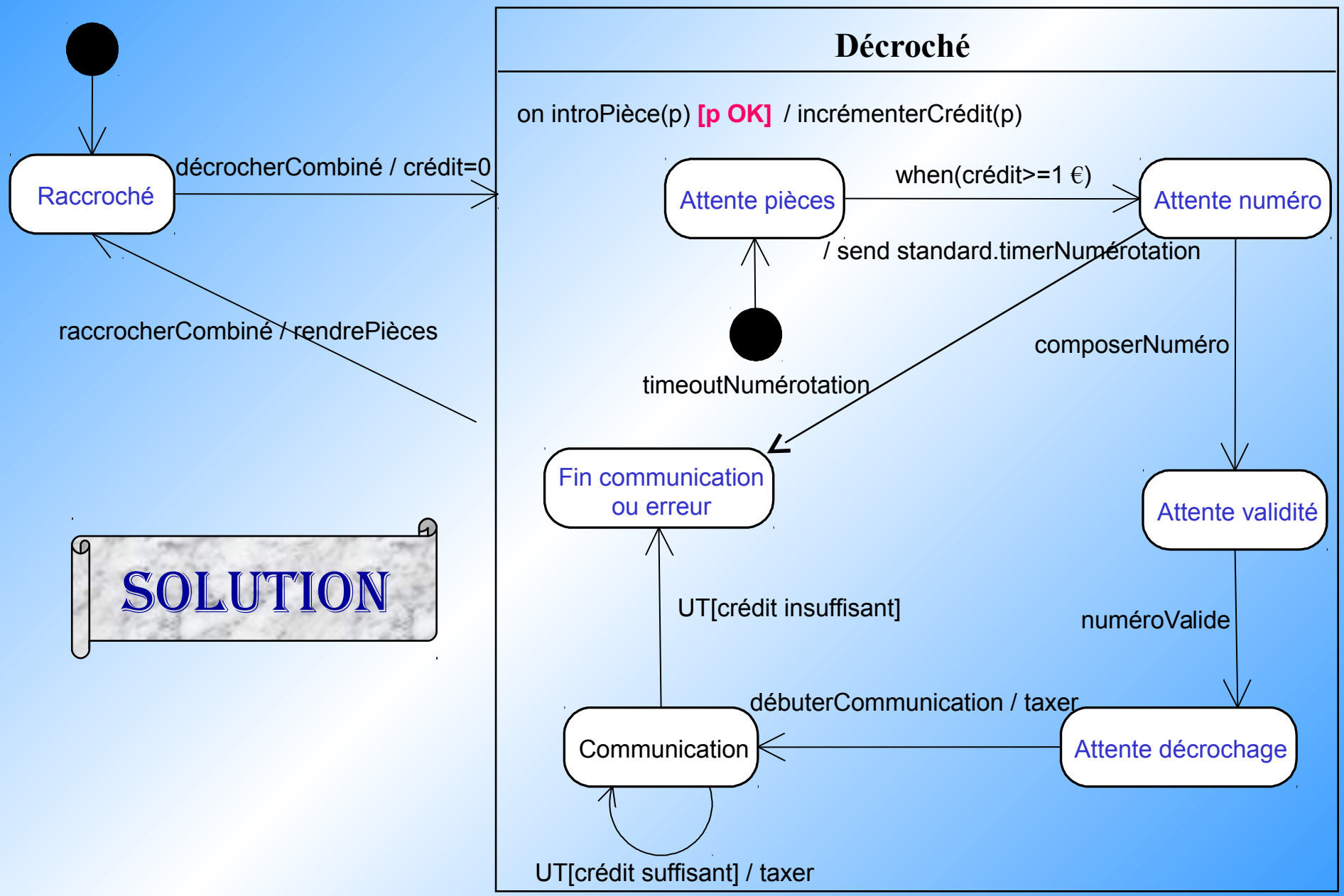

introPièce(p) / incrémenterCrédit(p)
Décroché
H
Raccroché
décrocherCombiné / crédit=0
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
raccrocherCombiné / rendrePièces
composerNuméro
Fin communication
Attente validité
SOLUTION
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer

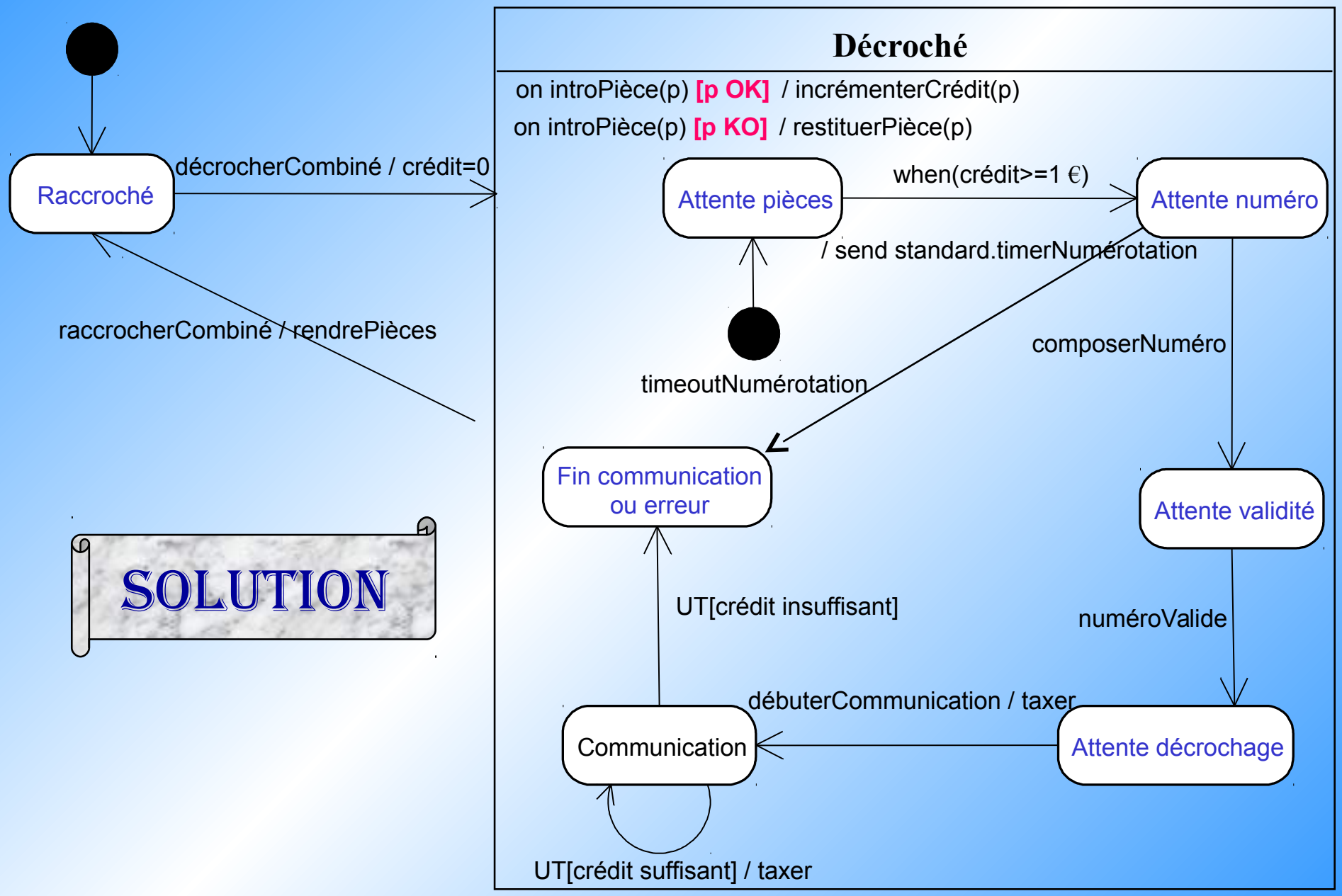
Phrase 2


introPièce(p) / incrémenterCrédit(p)
Décroché
H
Raccroché
décrocherCombiné / crédit=0
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
raccrocherCombiné / rendrePièces
timeoutNumérotation
composerNuméro
Fin communication
ou erreur
Attente validité
SOLUTION
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer

ÉTUDE DE CAS

Vérification de la validité de la pièce introduite


Décroché
on introPièce(p) [p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
timeoutNumérotation
composerNuméro
Fin communication ou erreur
Attente validité
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION


Décroché
on introPièce(p) [p OK] / incrémenterCrédit(p)
on introPièce(p) [p KO] / restituerPièce(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
timeoutNumérotation
composerNuméro
Fin communication ou erreur
Attente validité
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION

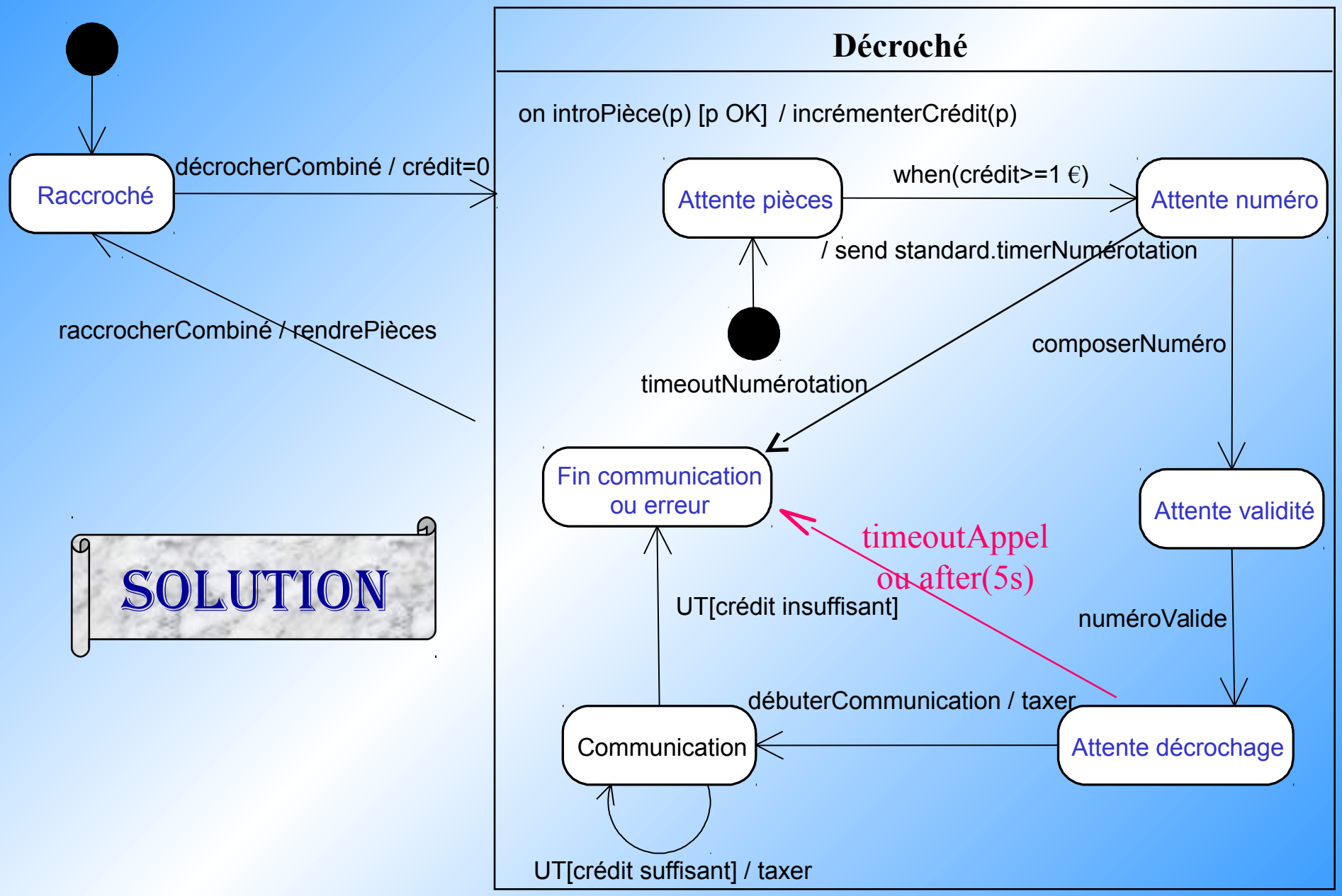
ÉTUDE DE CAS

Lorsque l'appelé ne décroche pas, la communication est interrompue

Étude de cas

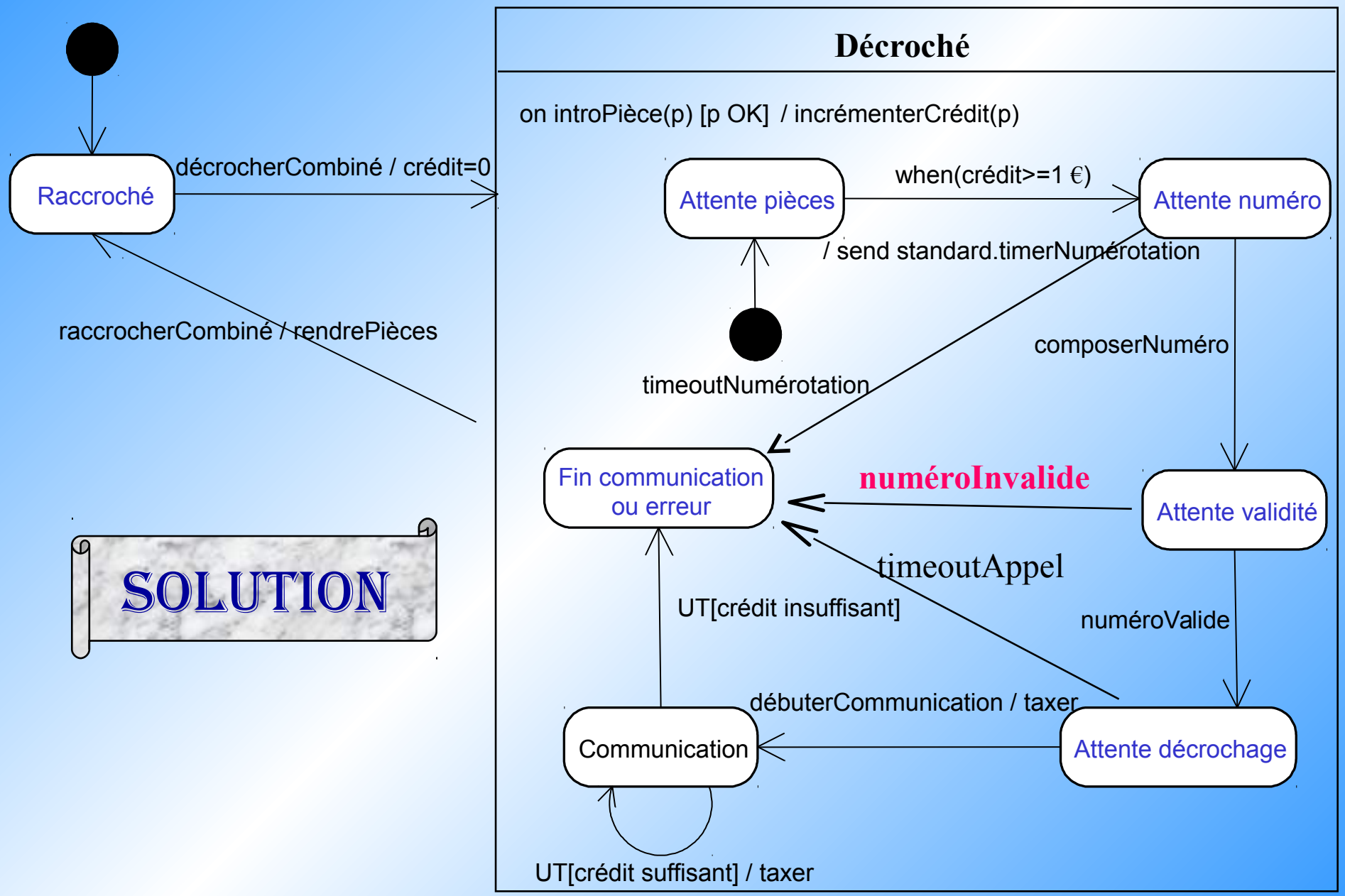
Lorsque l'appelé ne décroche pas, la communication est interrompue

Le standard envoie un message *timeoutAppel* au publiphone


Décroché
on introPièce(p) [p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
timeoutNumérotation
composerNuméro
Fin communication ou erreur
Attente validité
timeoutAppel ou after(5s)
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION

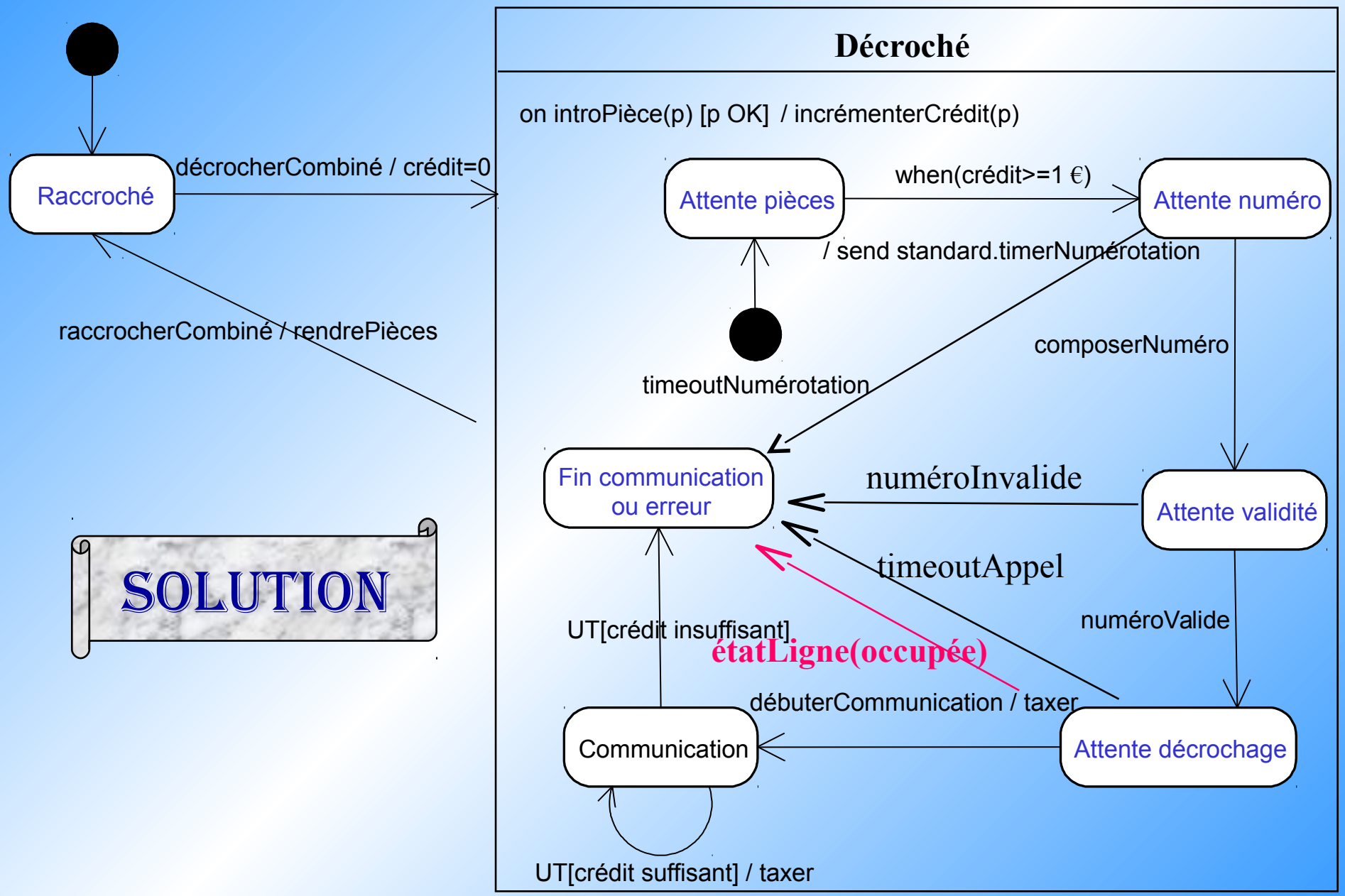
ÉTUDE DE CAS

Le numéro n'est pas valide


Décroché
on introPièce(p) [p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
timeoutNumérotation
composerNuméro
Fin communication ou erreur
numéroInvalide
Attente validité
timeoutAppel
UT[crédit insuffisant]
numéroValide
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION

Étude de cas

La ligne peut être libre ou occupée


Décroché
on introPièce(p) [p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
timeoutNumérotation
composerNuméro
Fin communication ou erreur
numéroInvalide
Attente validité
timeoutAppel
numéroValide
UT[crédit insuffisant]
étatLigne(occupée)
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION

SOLUTION

Phrase 3 : *La ligne peut être libre ou occupée.*

Pour le moment, on a supposé que la ligne était libre et que le correspondant décrochait.

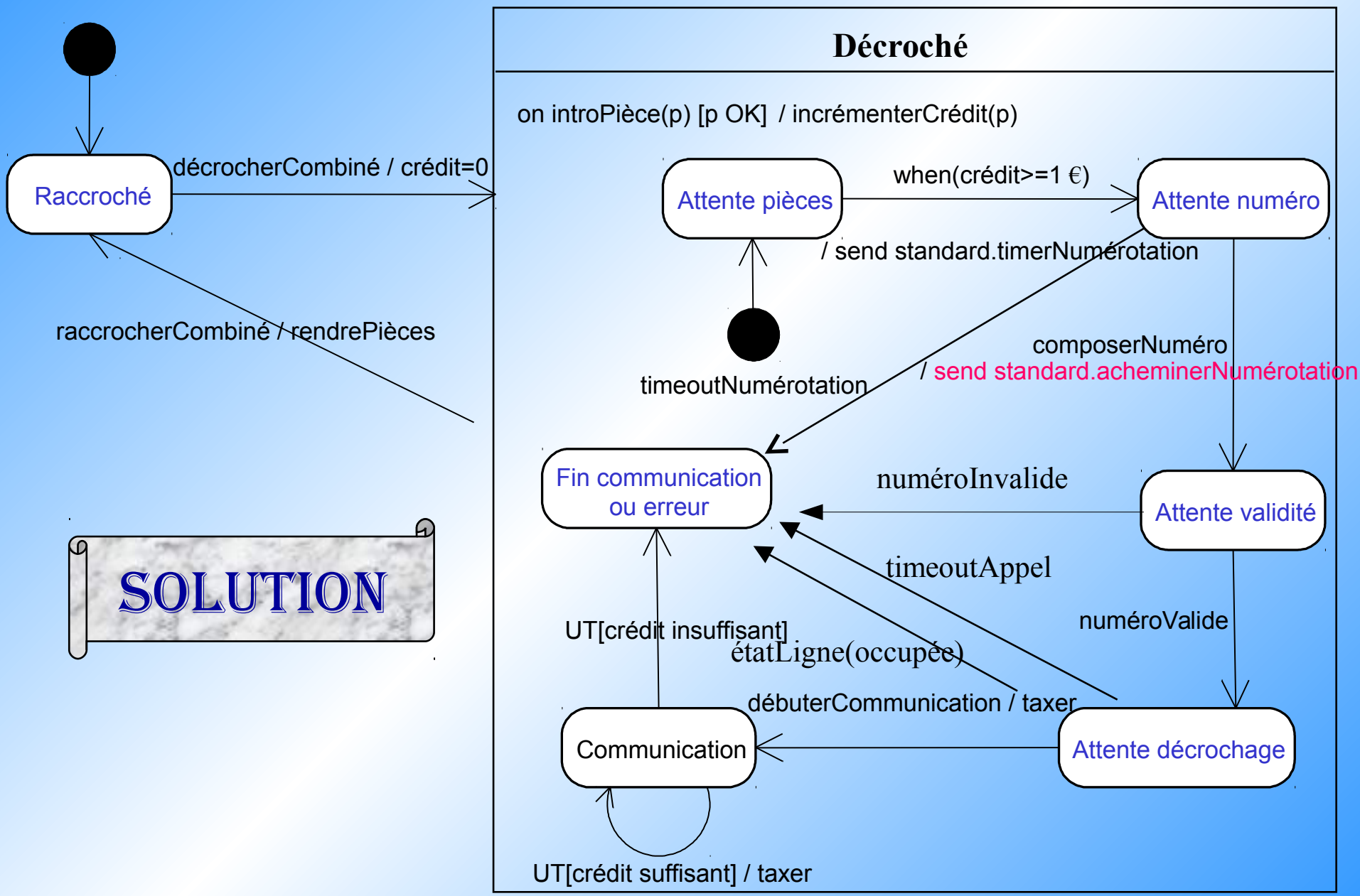
On va introduire dans le modèle :

- la possibilité que le standard renvoie un état ligne (*occupée*) (*étatLigne(occupée)*), mais également que

- l'appelé ne décroche pas (*ce qui n'est pas prévu explicitement dans l'énoncé*). Pour ce dernier cas, on suppose que le standard envoie un message *timeoutAppel* qui amène le Publiphone dans l'état d'erreur.

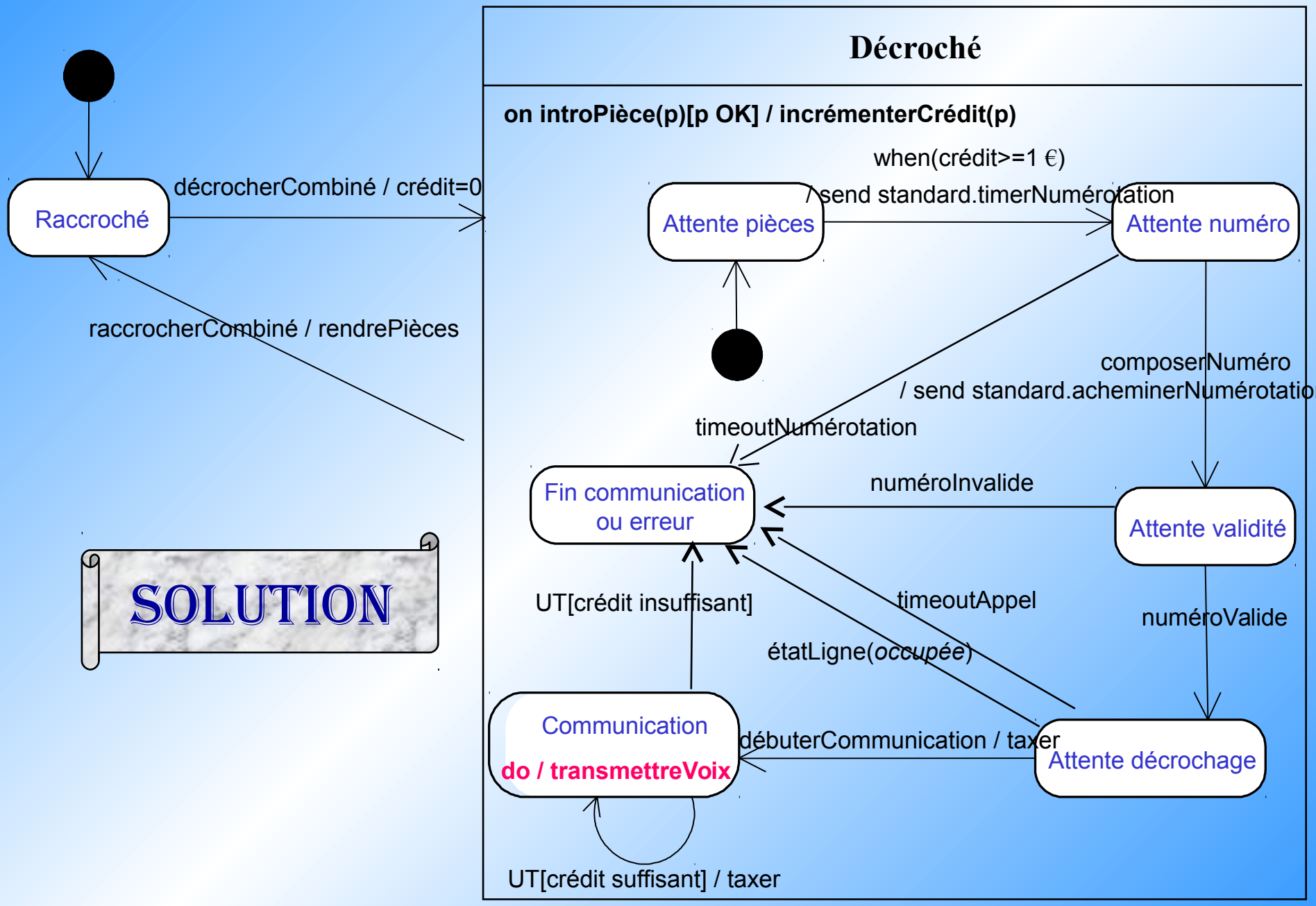
ÉTUDE DE CAS

Après la composition du numéro,
le publiphone achemine ce numéro au standard


Décroché
on introPièce(p) [p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send standard.timerNumérotation
timeoutNumérotation
composerNuméro
/ send standard.acheminerNumérotation
Fin communication
ou erreur
numéroInvalide
Attente validité
timeoutAppel
numéroValide
UT[crédit insuffisant]
étatLigne(occupée)
débuterCommunication / taxer
Communication
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION

Étude de cas

Y a-t-il une activité dans l'état communication ?


Décroché
on introPièce(p)[p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
when(crédit>=1 €)
/ send standard.timerNumérotation
Attente pièces
Attente numéro
composerNuméro
/ send standard.acheminerNumérotation
timeoutNumérotation
Fin communication
ou erreur
numéroInvalide
Attente validité
UT[crédit insuffisant]
timeoutAppel
numéroValide
étatLigne(occupée)
Communication
do / transmettreVoix
débuterCommunication / taxer
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION

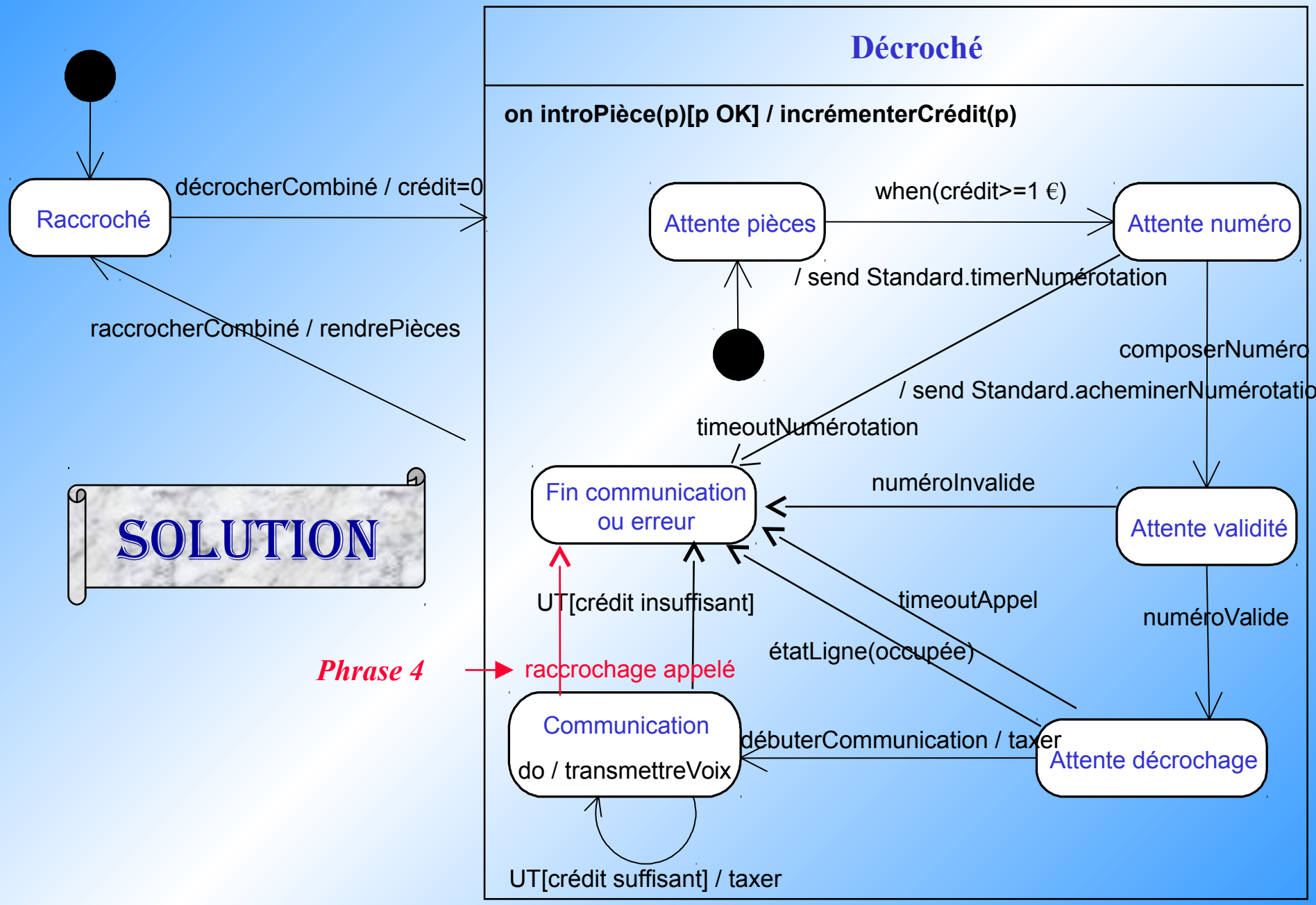
Étude de cas

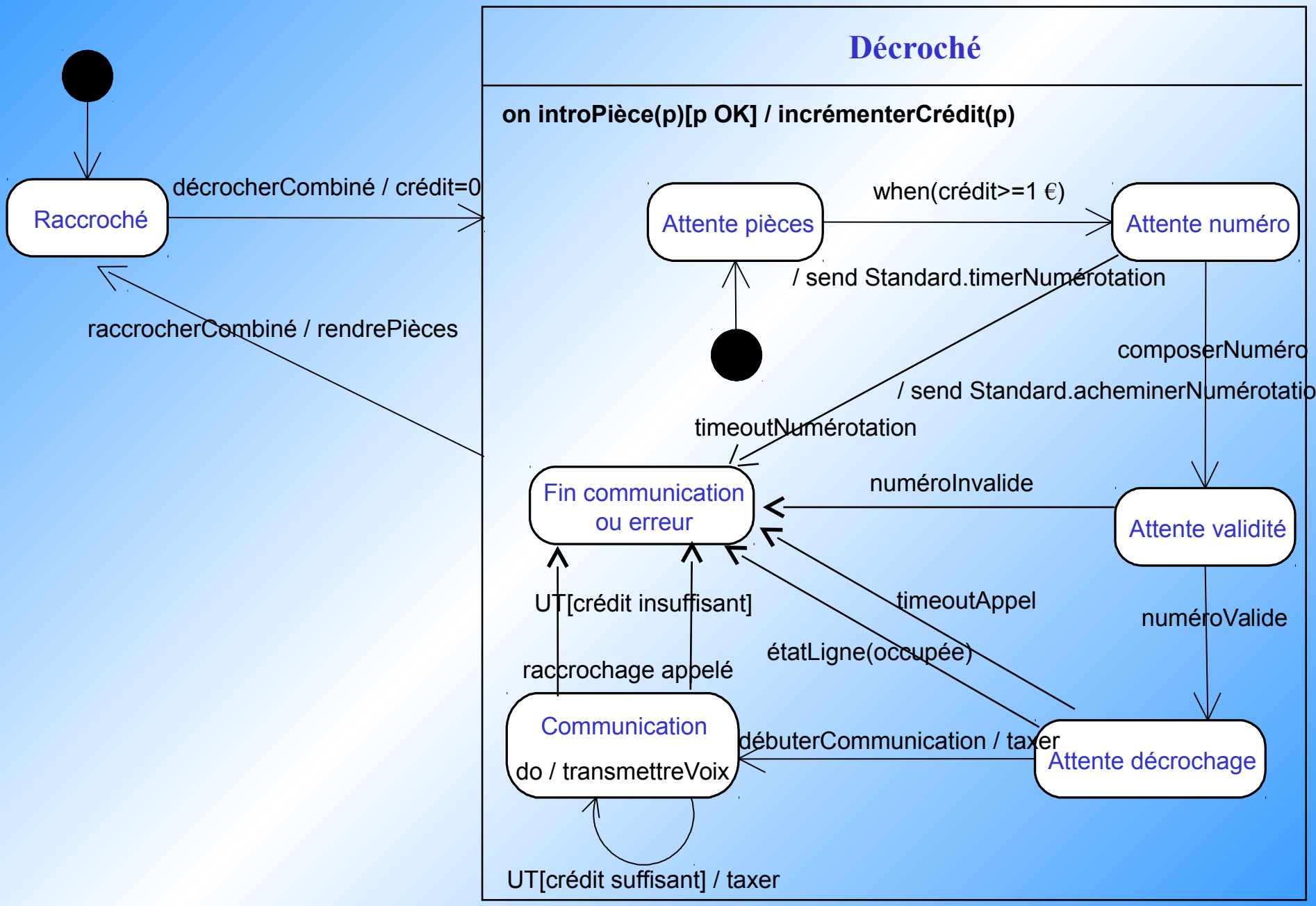
Le correspondant peut raccrocher le premier

SOLUTION

Phrase 4 *: le correspondant peut raccrocher le premier.*

La phrase *4* ajoute simplement une transition entre les états *Communication* et *Fin communication* (*raccrochage appelé*).


Décroché
on introPièce(p)[p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send Standard.timerNumérotation
composerNuméro
/ send Standard.acheminerNumérotation
timeoutNumérotation
Fin communication ou erreur
numéroInvalide
Attente validité
UT[crédit insuffisant]
timeoutAppel
numéroValide
étatLigne(occupée)
Phrase 4
raccrochage appelé
Communication
do / transmettreVoix
débuterCommunication / taxer
Attente décrochage
UT[crédit suffisant] / taxer
SOLUTION


Décroché
on introPièce(p)[p OK] / incrémenterCrédit(p)
Raccroché
décrocherCombiné / crédit=0
raccrocherCombiné / rendrePièces
Attente pièces
when(crédit>=1 €)
Attente numéro
/ send Standard.timerNumérotation
composerNuméro
/ send Standard.acheminerNumérotation
timeoutNumérotation
Fin communication ou erreur
numéroInvalide
Attente validité
UT[crédit insuffisant]
timeoutAppel
numéroValide
étatLigne(occupée)
raccrochage appelé
Communication
do / transmettreVoix
débuterCommunication / taxer
Attente décrochage
UT[crédit suffisant] / taxer

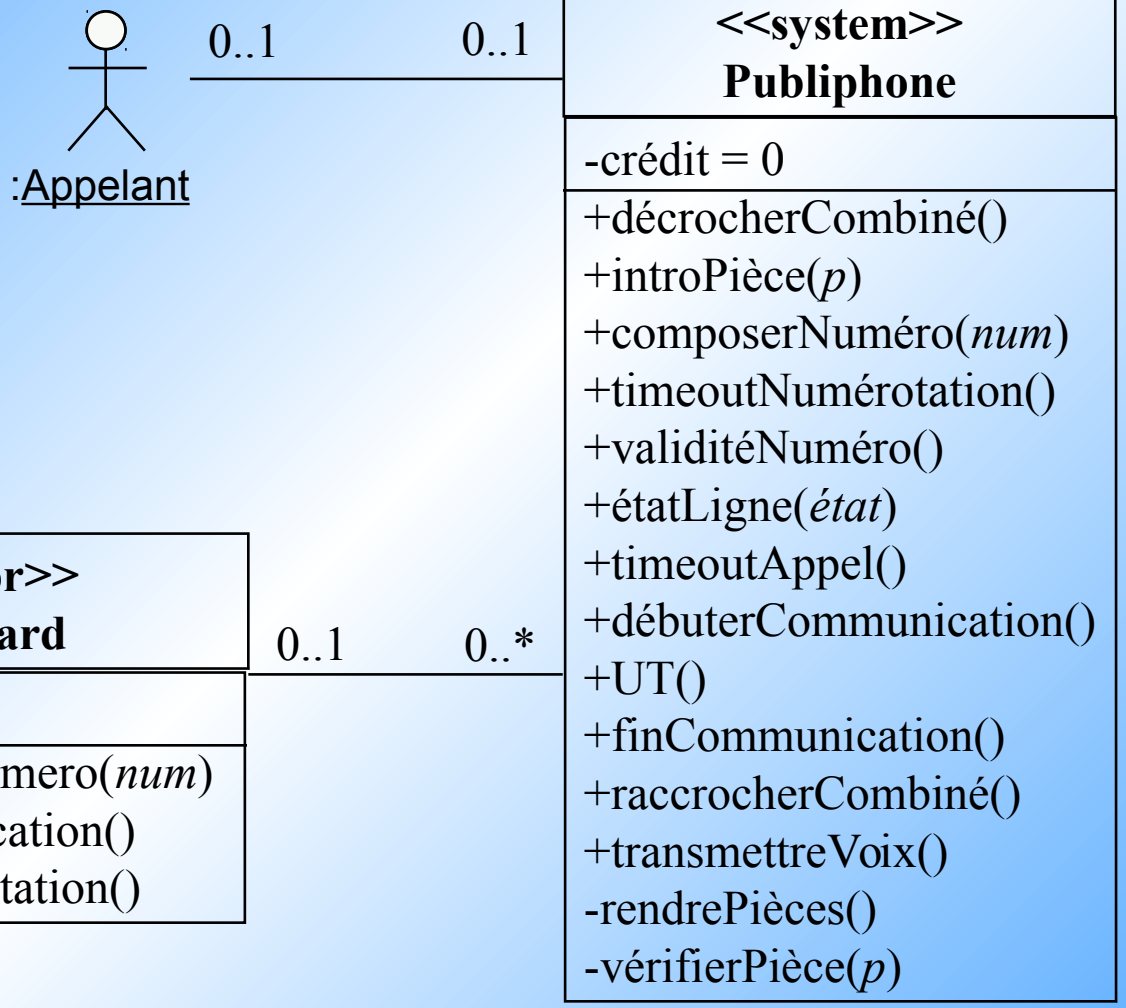
Étude de cas

En vous servant de toute cette étude dynamique, proposez une version étendue du diagramme de contexte statique qui fasse apparaître les attributs et les opérations de la classe Publiphone.

Diagramme de contexte statique étendu

On appelle **diagramme de contexte statique étendu**, un diagramme de contexte statique dans lequel on ajoute des attributs et des opérations de niveau système à la classe qui représente le système (*appréhendé comme une boite noire*), ainsi qu'aux acteurs non-humains.

SOLUTION


:Appelant
0..1
0..1
<<system>>
Publiphone
-crédit = 0
+décrocherCombiné()
+introPièce(p)
+composerNuméro(num)
+timeoutNumérotation()
+validitéNuméro()
+étatLigne(état)
+timeoutAppel()
+débuterCommunication()
+UT()
+finCommunication()
+raccrocherCombiné()
+transmettreVoix()
-rendrePièces()
-vérifierPièce(p)
<<actor>>
Standard
+acheminerNumero(num)
+finCommunication()
+timerNumérotation()
0..1
0..*