Conception Orientée Objet

Diagramme de séquence

Tianxiao LIU
Master IISC 1ère Année
CY Cergy Paris Université
http://depinfo.u-cergy.fr/~tliu/coo.php

Plan

- Motivation
- Notions de base
  - Ligne de vie et messages
- **Fragments combinés**
  - Choix et boucle
  - Contrôle d'envoi en parallèle de message
  - Fixer l'ordre d'envoi des messages
  - Interprétation des fragments
- Un exemple

Motivation

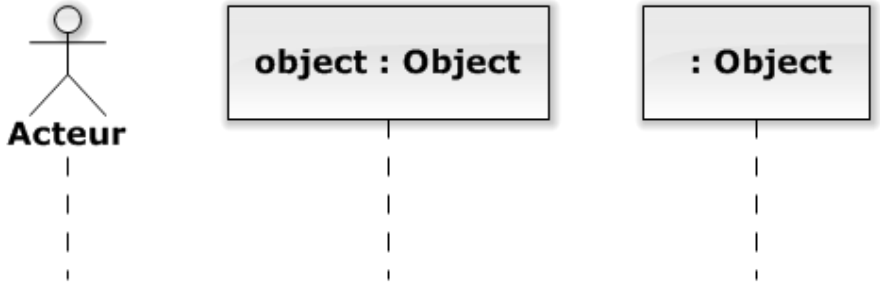
- Diagramme de cas d'utilisation
  - Acteurs interagissant avec les grandes fonctionnalités d'un système
  - Vision fonctionnelle et externe d'un système

- Diagramme de classe
  - Les classes et la façon dont elles sont associées
  - Vision statique et structurelle d'un système

- **Diagramme de séquence**
  - Un **pont** entre les deux approches ci-dessus
  - Interactions entre les instances au cœur du système pour **réaliser une certaine fonctionnalité**

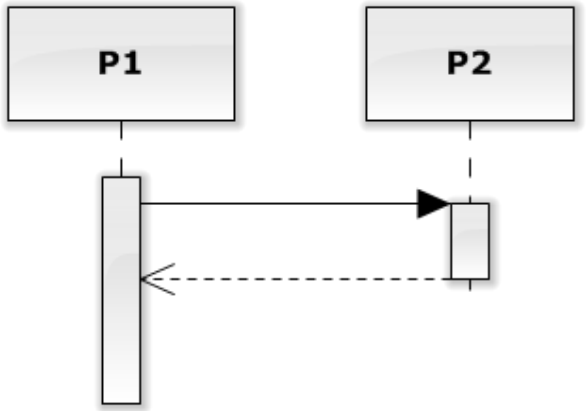
Ligne de vie

- Principe
  - Un participant dans l'interaction
  - Syntaxe du nom de la ligne de vie
    - **(nomObjet) : nomClasse**
- Participants possibles
  - Une **instance** d'une classe
  - Un **composant** du système
  - Un **acteur** externe



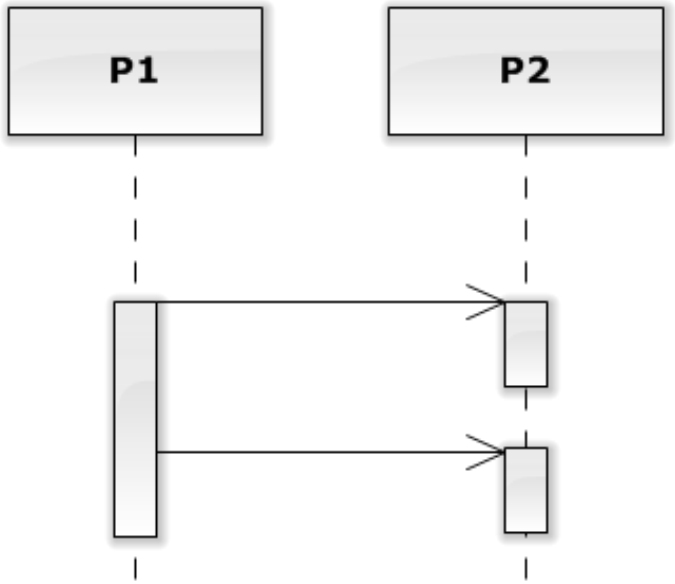
Les messages

- Principe de base
  - Messages échangés entre les lignes de vie
  - Un message = Un communication particulière entre les lignes de vies
- **Occurrence d'exécution**
  - Réaction déclenchée par réception d'un message
  - Exécution d'une méthode d'une classe



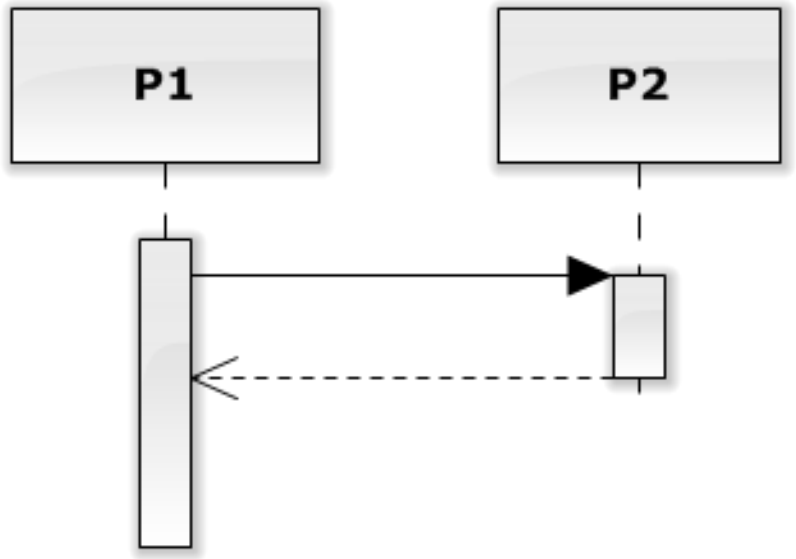
Les messages

- Messages **asynchrones**
  - L'émetteur ne reste pas bloqué et continue son exécution
  - Les messages asynchrones peuvent être reçus dans un ordre différent de l'ordre d'envoi.



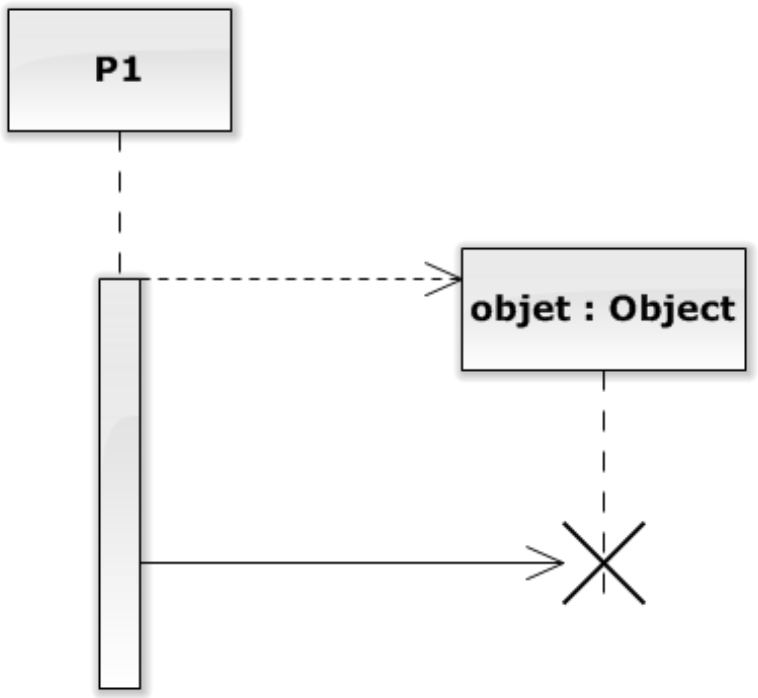
Les messages

- Messages **synchrones**
  - L'émetteur du message reste **bloqué** le temps que le récepteur traite le message et renvoie une réponse Message de retour



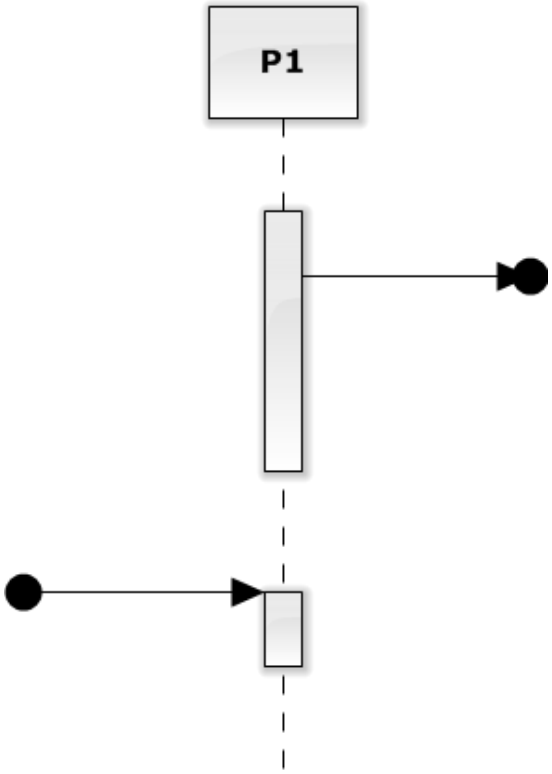
Les messages

- Message de création d'un objet
- Message de destruction d'un objet



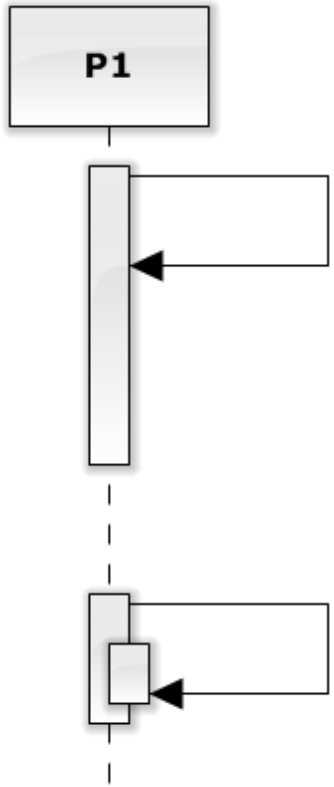
Les messages

- Message complet : envoi et réception connus
- Message perdu : envoi connu mais pas réception
- Message trouvé : réception connue mais pas envoi



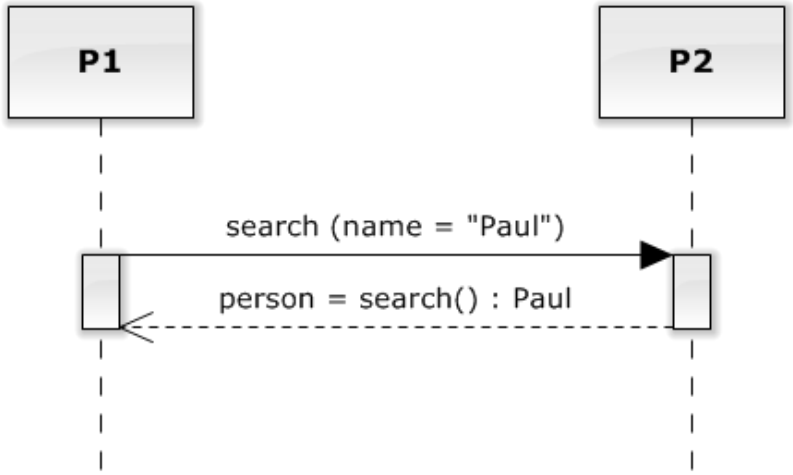
Les messages

- Message réflexif : une activité interne de l'objet
- Message récurrent : un message envoyé à objet lui-même



Les messages

- Syntaxe des messages
  - Message d'envoie
    - **nomMessage (arguments)**
    - nomMessage est un signal ou une opération
    - Pour un argument : **nomParamètre (= valeur)**
  - Message de retour
    - **attribut = (messageEnvoi : valeurRetour)**



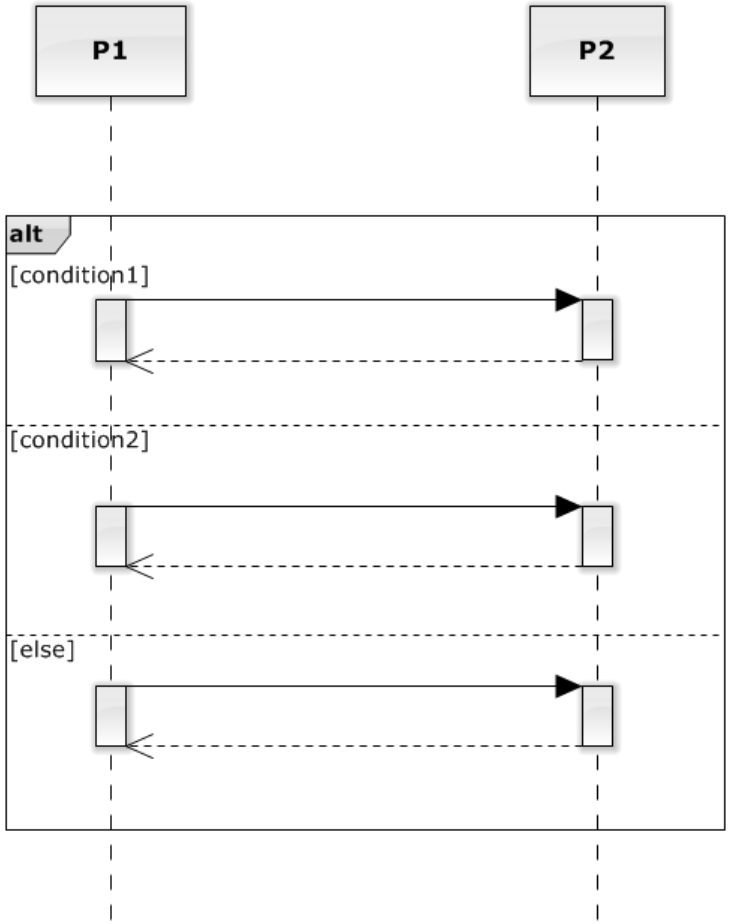
Fragments combinés

- Objectifs
  - Décomposer un système complexe en fragments suffisamment simples
  - Restituer la complexité du système
  - **Combiner les fragments**

- **Opérateurs** pour fragments combinés
  - Choix et boucle : *option, alternative, loop, break*
  - Parallélisme : *parallel, critical region*
  - Fixer l'ordre d'envoi des messages : *strict sequencing, weak sequencing*
  - Interprétation des fragments : *ignore, consider, assertion, négative*

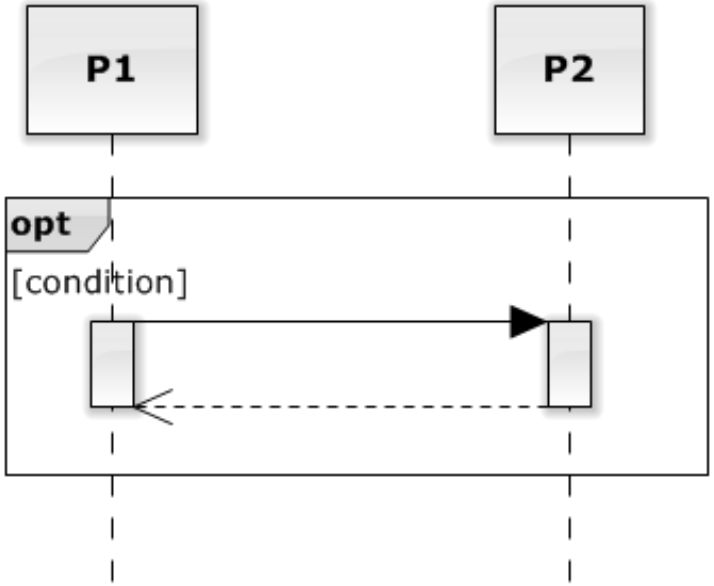
Fragments combinés

- *Alternative* (**alt**)
  - Contient une liste d'opérandes alternatifs
  - Un seul **opérande** sera choisi
  - Possibilité de mettre des conditions
  - Possibilité d'utiliser ***else***



Fragments combinés

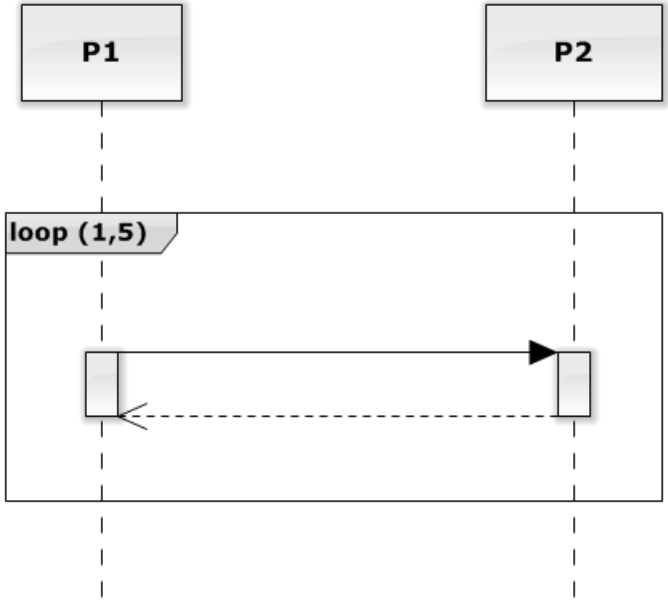
- *Option* (**opt**)
  - Equivalent à une alternative ayant le choix **entre un seul opérande et rien**



Fragments combinés

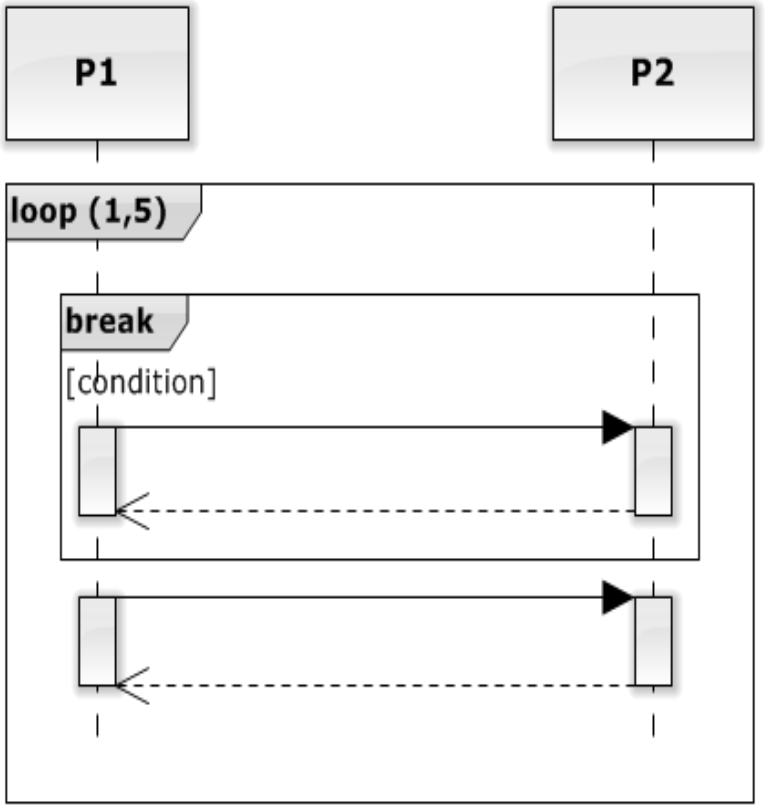
- *Loop* (**loop**)
  - Répétition de l'opérande
  - Contrôle de l'itération
    - loop (min, max)
    - loop : nombre de répétitions indéfini
    - loop (valeur) →

      loop (valeur, valeur)
    - On peut aussi utiliser une condition



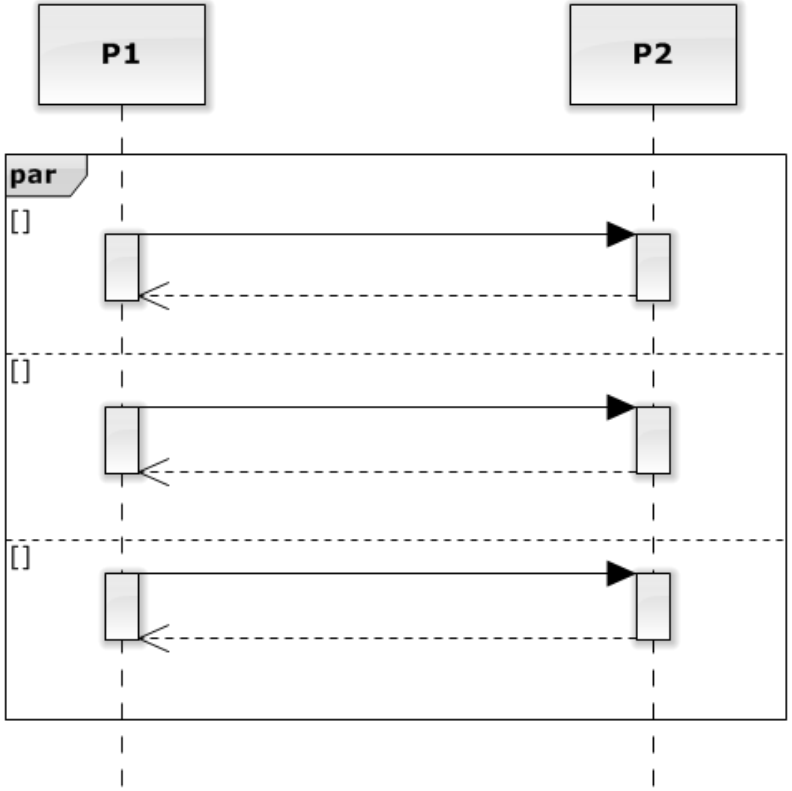
Fragments combinés

- *Break* (**break**)
  - Utilisé dans un autre fragment
    - Exécuter le break fragment
    - Interrompre l'exécution du reste du fragment
  - Possibilité de mettre une condition



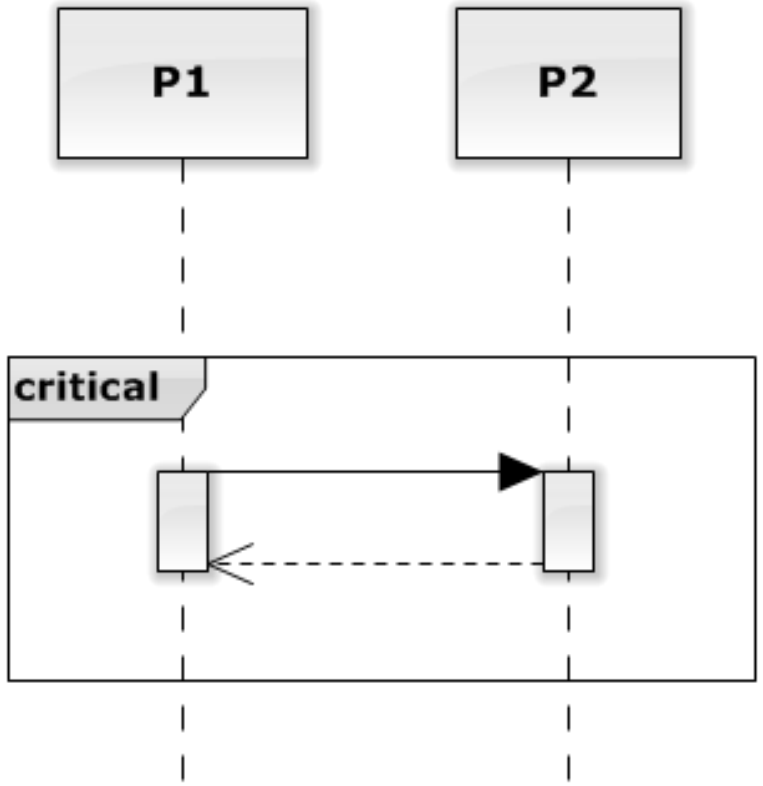
Fragments combinés

- *Parallel* (**par**)
  - Exécution potentiellement en parallèle des opérandes



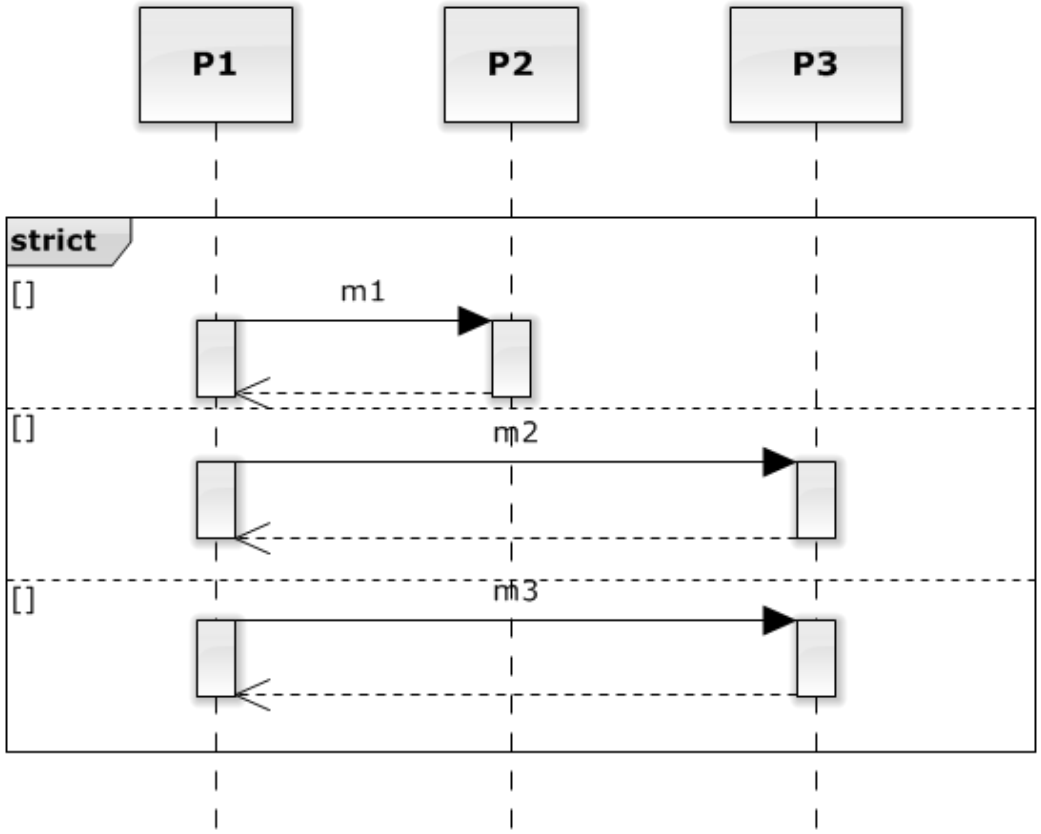
Fragments combinés

- *Critical Region* (**critical**)
  - Définir une section critique
  - Les interactions dans ce fragment **ne peuvent pas être interrompues (ou s'entrelacer)** par (avec) d'autres interactions dans le diagramme
  - Un traitement **atomique**



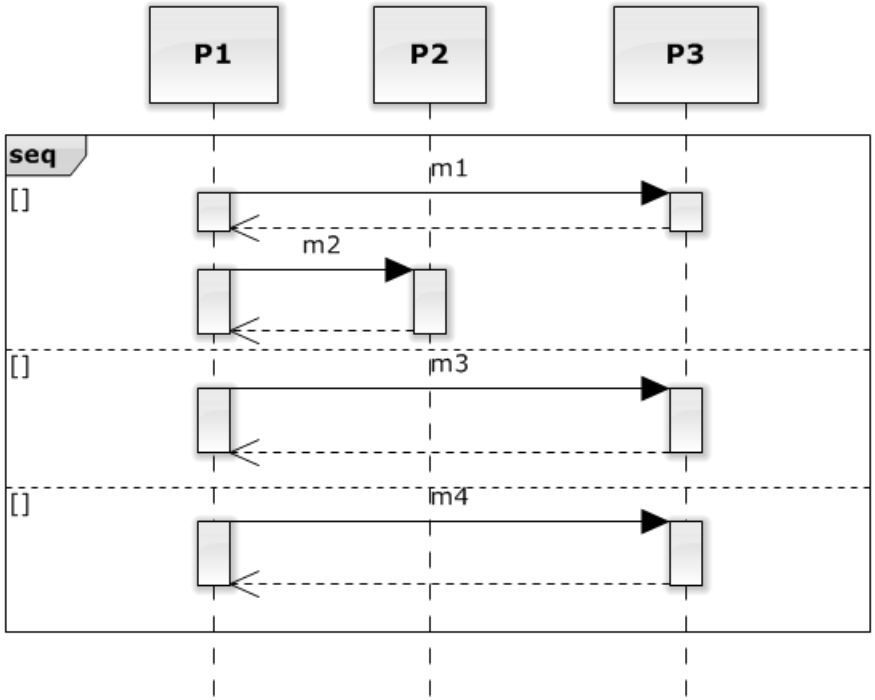
Fragments combinés

- *Strict sequencing* (**strict**)
  - Imposer l'ordre des messages décrits dans **tous** les opérandes



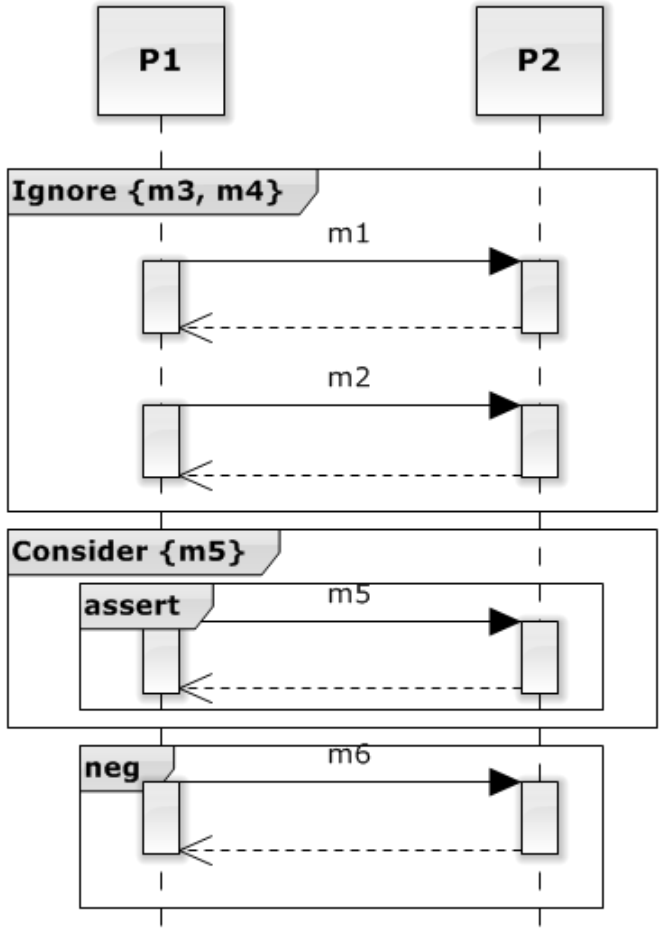
Fragments combinés

- *Weak sequencing* (**seq**)
  - **Imposer** l'ordre des messages décrits dans chaque opérande
  - **Imposer** l'ordre des messages concernant les mêmes lignes de vie dans différents opérandes
  - Pas d'ordre imposé pour les messages concernant différentes lignes de vie dans différents opérandes



Fragments combinés

- *Ignore, consider, assertion, negative*
  - Permet de mieux interpréter le diagramme



m3 et m4 peuvent avoir lieu, mais ils ont aucune importance pour le diagramme.

Il peut y avoir autres messages que m5, mais ils sont ignorés. Et m5 doit avoir lieu

Si m6 a lieu, le système échoue.

Un exemple concret : un robot