05 Декабря2013

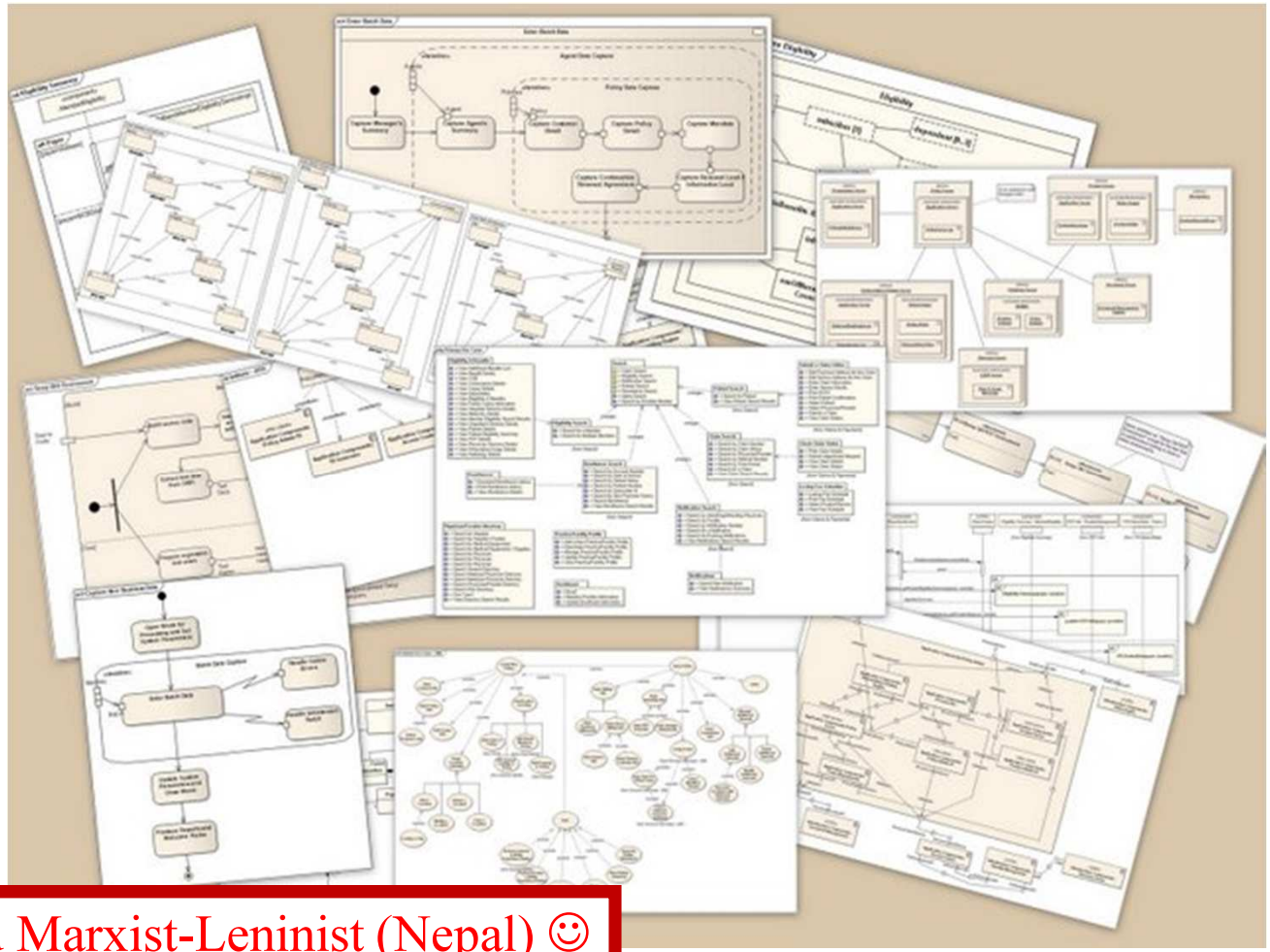
Инсерционное моделирование 1

Лекция 14

Графические модели

UML languages

Object Management Group, object-oriented engineering



UML: United Marxist-Leninist (Nepal) ☺

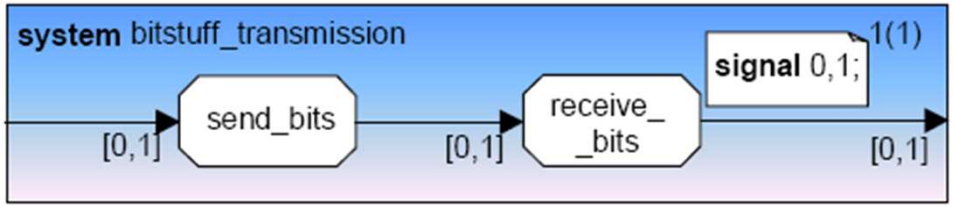
ITU (International Communication Union) languages

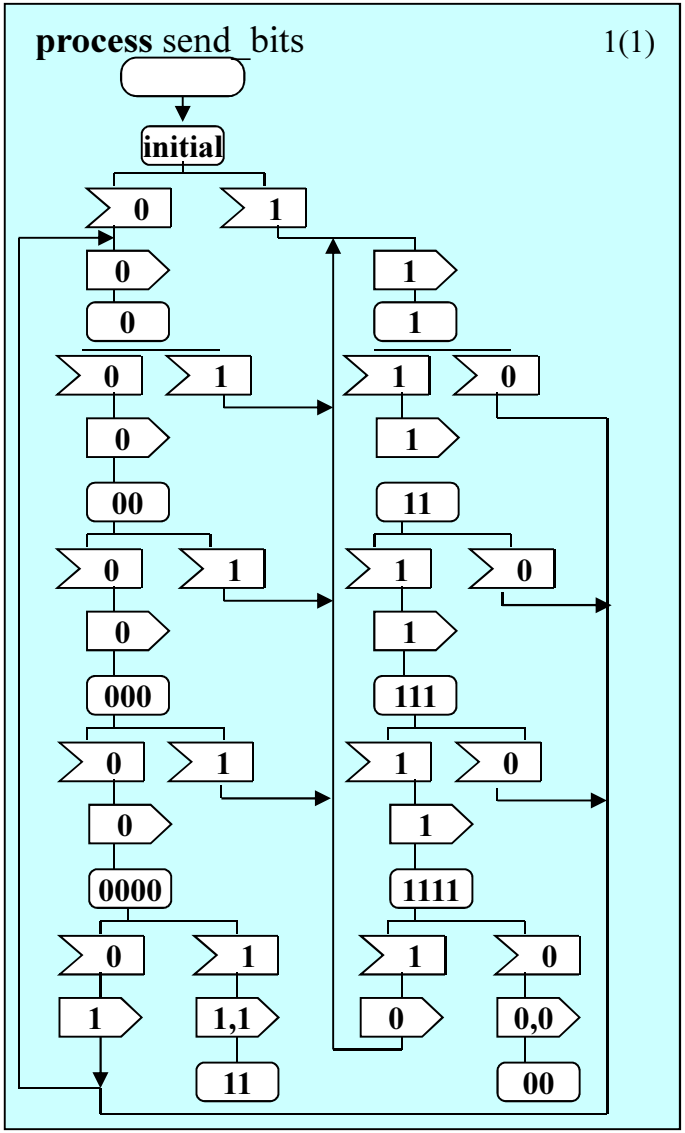
SDL, MSC, UCM
Behavioral languages

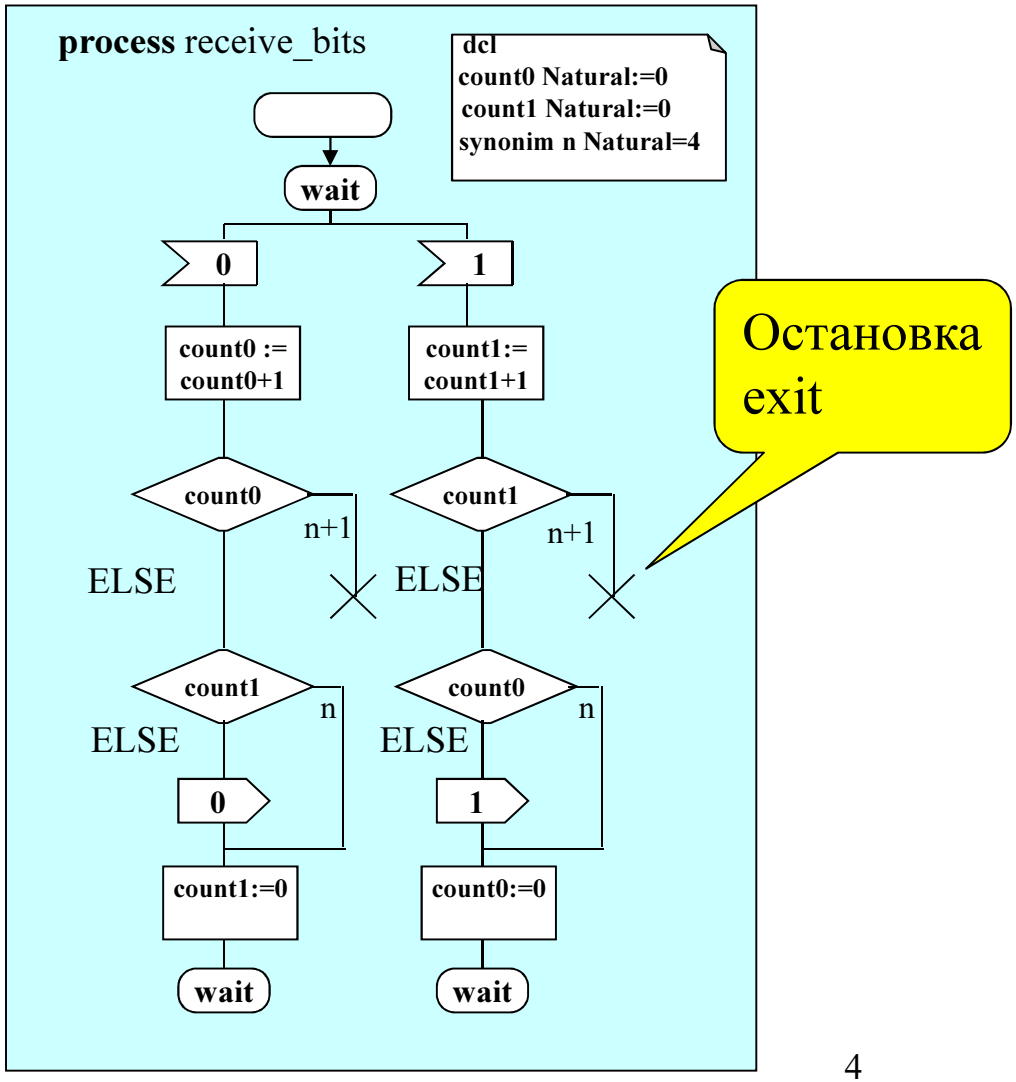
Convergence with UML
Use cases, sequencing diagrams, State machines

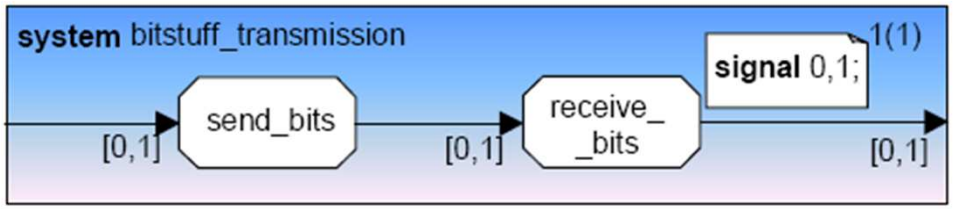
SDL

Процессы и блоки









Example simple system model **Bit-stuffing one-way transmission**.
This system consists of a send-bits transmitter and a receive-bits receiver. The transmitter inserts (stuffs in) bits so that there are never n bits the same. The receiver removes the inserted bits.

This technique is used in real systems to protect against stuck at zero or one or (for example in Signaling System 7) to allow flags that consist of n ones or zeros to be inserted without the risk that they are imitated by signals.

Пример простой модели системы **Bit-stuffing one-way transmission**
(односторонняя передача с битовым наполнением).
Эта система состоит из передатчика send-bits и приемника receive-bits. Передатчик вставляет биты, так что бы не было n одинаковых битов подряд. Приемник удаляет вставленные биты.

Этот метод используется в реальных системах для защиты от повторения нулей и единиц (например, в Signaling System 7), чтобы можно было использовать разделительные флаги, которые состоят из n единиц или нулей, без риска, что они смешаются с сигналами.

Задание: Доказать, что на выходе системы будет то же самое, что и на входе

Инсерционная модель

Уровень блоков
Среда: сеть, связывающая блоки буферизованными каналами
Агенты: процессы или блоки
Прием сообщений
Передача сообщений
Параллельное погружение
Много агентов

Уровень процессов
Среда: память
Агенты:
Условия, присваивания, передача сообщений прием сообщений
Последовательное погружение
Один агент

Функция погружения

$$\frac{E \xrightarrow{a} E', u \xrightarrow{a} u'}{E[u] \longrightarrow E'[u']}$$

a – внутреннее действие

$$\frac{E \xrightarrow{a} E', u \xrightarrow{a} u'}{E[u] \xrightarrow{a} E'[u']}$$

a – внешнее действие

Внутренние действия процессов: условия и присваивания
Внешние действия процессов: обмен сообщениями
Внутренние действия блоков: обмен по внутренним каналам
Внешние действия блоков: обмен по внешним каналам (входным-выходным)

Погружение нескольких агентов

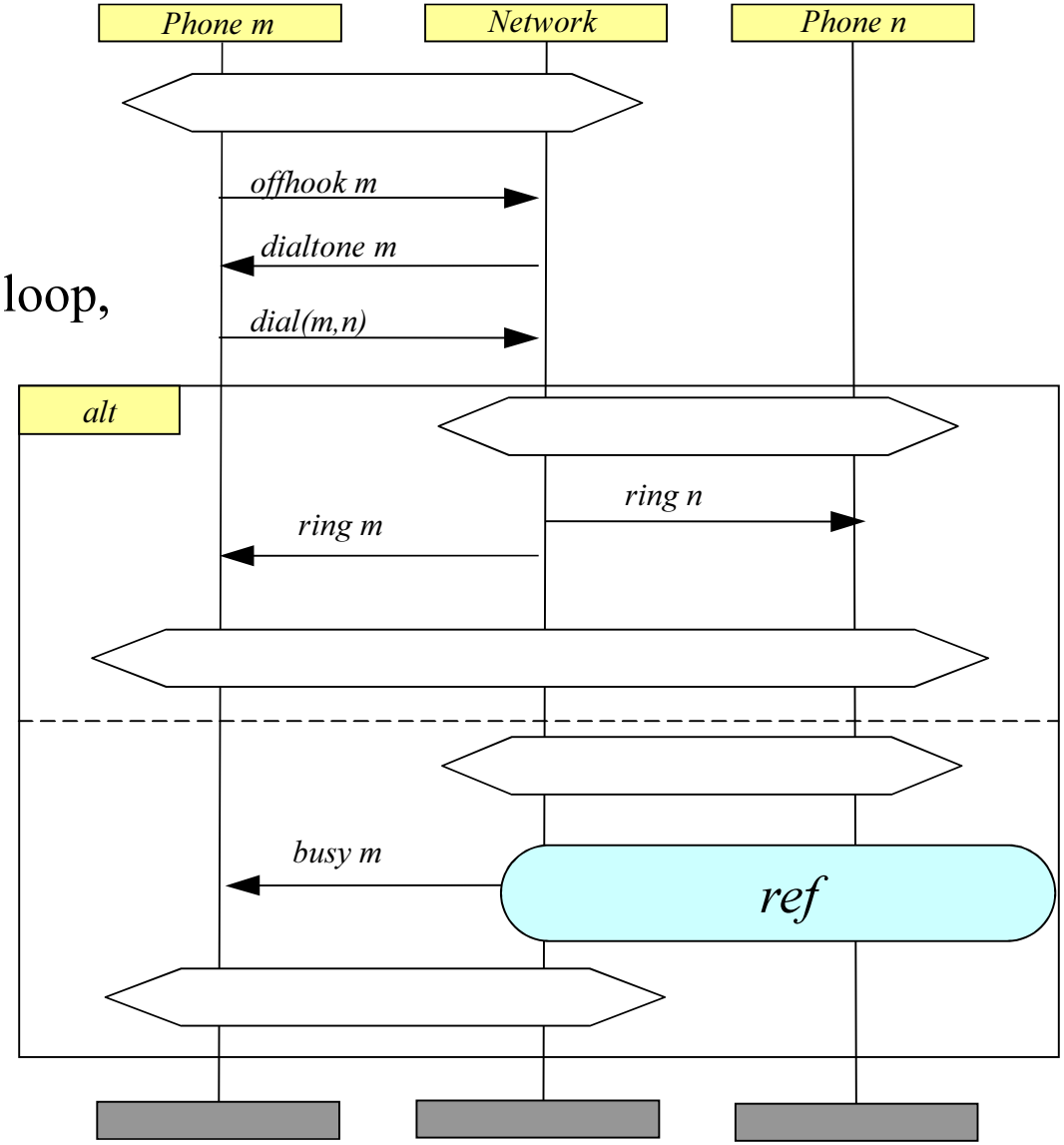
$$E[u_1, u_2, \ldots] = E[u_1 \parallel u_2 \parallel \ldots]$$

Параллельная композиция: чистый интерливинг
Имена агентов: передаются через действия

$$u \xrightarrow{a} u' \Rightarrow m : u \xrightarrow{m:a} m : u'$$

MSC

opt, exc
par, seq, loop,



MSC агенты-инстанции

Действия

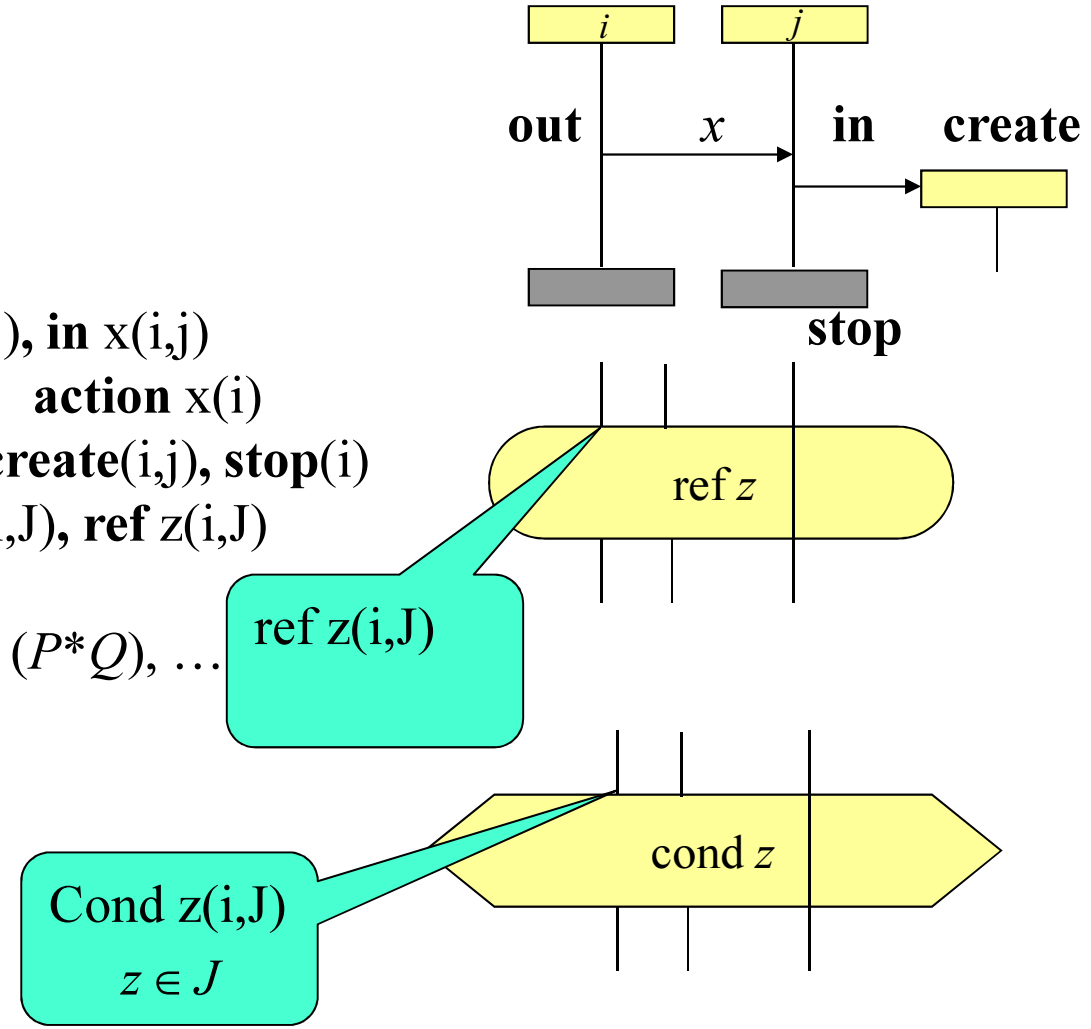
Сообщения: **out** x(i,j), **in** x(i,j)

Локальные действия: **action** x(i)

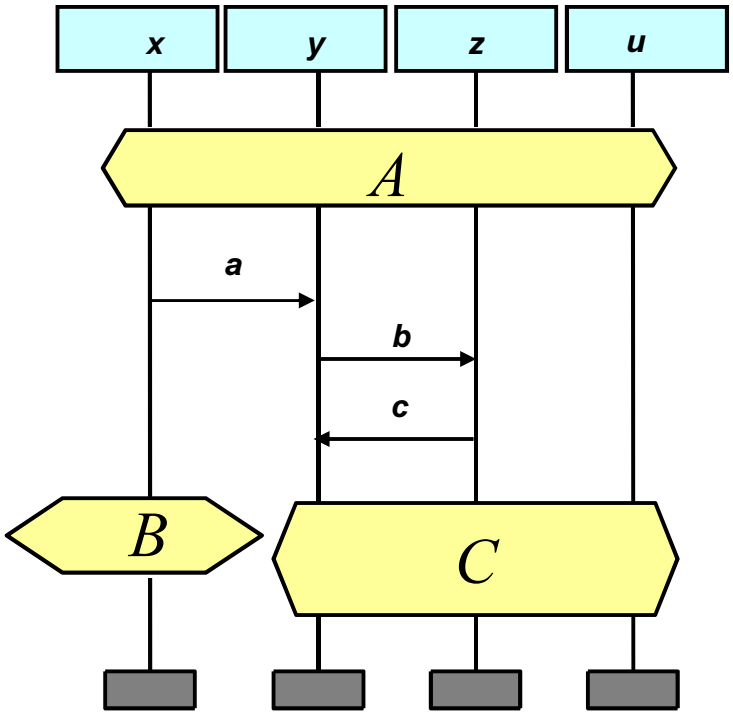
Инстанции: **inst**(i), **create**(i,j), **stop**(i)

Управление: **cond** z(i,J), **ref** z(i,J)

Functions: $(P;Q)$, $(P||Q)$, (P^*Q), …



MSC диаграмма => MSC агент



inst(x).cond A(x,J).out(a,x,y).cond B(x,{x}).stop x | |
inst(y).cond A(y,J). in(a,x,y).out(b,y,z). in(c,z,y).
cond C(y,K).stop y | |
inst(z).cond A(z,J). in(b,y,z).out(c,z,y).
cond C(z,K).stop z | |
inst(u).cond A(u,J).cond C(u,K).stop u

J={x,y,z,u}, K={y,z,u}

on-line выражения определяются с помощью функциональных выражений и недетерминированного выбора (статья в Кибернетике).

Правильный порядок:

out =>in

Синхронизация по условиям

За порядком следит среда

MSC среда

Состояние среды разрешает или запрещает выполнение действий в соответствии с их частичным порядком и синхронизацией определяемой условиями и ссылками.

Разрешающая аддитивная функция погружения

$$\frac{e \xrightarrow{a} e'[v], u \xrightarrow{a} u'}{e[u] \xrightarrow{a} e'[v * u']} P(e,a)$$

* слабая последовательная (параллельно-последовательная композиция

Среда

Состояние : трасса, последовательность выполненных действий

$$a_1, a_2, \ldots, a_n, \ldots$$

$$(a_1, a_2, \ldots, a_n) \xrightarrow{a_{n+1}} (a_1, a_2, \ldots, a_n, a_{n+1})$$

Инстанция i открыта в точке a_n: было выполнено inst(i) и не было stop(i)
Инстанция i заблокирована в точке a_n : $i \in J$, было выполнено cond(i,J), но не для всех $k \in J$ было выполнено cond(k,J).
Аналогично для действий ref(i,J).
Действие out x(i,j), находится в состоянии ожидания, если количество предшествующих действий out x(i,j) не превышает количества предшествующих действий in x(i,j).

Разрешающая функция

a	$P(e,a)$
out x(i,j),	инстанции i и j открыты и i не заблокирована
in x(i,j)	инстанции i и j открыты, **out** находится в состоянии ожидания, j не заблокирована
action x(i)	инстанция i открыта и не заблокирована
inst(i),	инстанция i не открыта
create(i,j),	i открыта и не заблокирована, j не открыта
stop(i)	i открыта и не заблокирована
cond z(i,J),	инстанция i открыта и не заблокирована
ref z(i,J)	инстанция i открыта и не заблокирована

Упражнение

Написать регулярные выражения (уравнения) для определения свойств открытости, заблокированности и ожидания.
Формализовать разрешающую функцию

Погружение нового агента

$$\frac{e \xrightarrow{a} e'[v], u \xrightarrow{a} u'}{e[u] \xrightarrow{a} e'[v * u']} P(e, a)$$

a=**ref** z(i, J), все инстанции из J не заблокированны
v=val(z)

Слабая последовательная композиция MSC-агентов

$$e[P,Q] = (e[P])[Q] = e[P * Q]$$

$$P = p_1 \,||\, ... \,||\, p_m \,||\, q_1 \,||\, ... \,||\, q_k$$
$$Q = r_1 \,||\, ... \,||\, r_l \,||\, q'_1 \,||\, ... \,||\, q'_k$$
$$P * Q = p_1 \,||\, ... \,||\, p_m \,||\, (q_1 ; q'_1) \,||\, ... \,||\, (q_k ; q'_k) \,||\, r_1 \,||\, ... \,||\, r_l$$

$p_1, ..., p_m, r_1, ... r_l$ — *различные инстанции*

q_i, q'_i — *одинаковые инстанции*

$$(\sum P_i) * Q = \sum (P_i * Q)$$

Вложенные выражения
(on-line expressions)

F – функция перевода в язык процессов

F(**loop**(m,n) E) = **loop**(m,n,F(E));
F(**alt** E1 **alt** E2 **alt**...)= F(E1) + F(E2)+...;
F(**opt** E) = F(E) +Δ;
F(**par** E1 **par** E2 **par**...) = F(E1) | | F(E2) | | ... ;
F(**seq** E1 **seq** E2 **seq**...) = (F(E1); F(E2);...);
F(**exc** E) = (F(E);0) + Δ;
F(P) = P;

loop(0,0,G) = Δ,
loop(0,inf,G) = (G***loop**(0,inf,G)) + Δ,
loop(m,inf,G) = (G***loop**(m-1,inf,G)),
loop(0,n,G) = (G***loop**(0,n-1,G)) + Δ,
loop(m,n,G) = (G***loop**(m-1,n-1,G)).

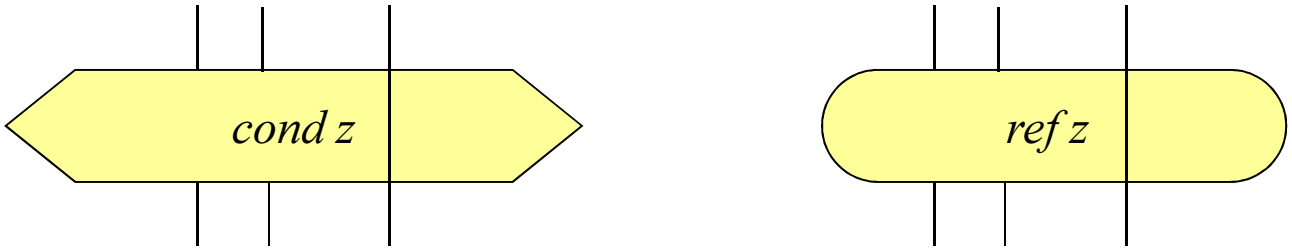
Другие варианты состояния среды

- ***Состояние среды:***
  - *история*
  - *формула линейной темпоральной логики*
  - *состояние автомата* (*регулярные выражения*)
  - *простой объект* (*атрибутная среда*)

MSC среда простой объект

Состояние

***<process:**U, **out:**O, **synchr:**S, **perform:**P, **allinst:**F>*



$U : names \to MSCs$

$O(x,i,j) = m \Leftrightarrow$ *m событий* ***out*** *x(i,j) не имеют соответствующих* ***in-ов***
(находятся в состоянии ожидания)

$S(x,y,J) = I \subseteq J$, *y =* ***wait cond*** *z,* ***wait ref*** *z, x – имя ссылки*
(инстанции J **заблокированы, пока** $J \setminus I \neq \varnothing$ **)**

$P(z) = (J,y) \Leftrightarrow$ *ссылка z присоединенная к J выполняется внутри ссылки y* **(инстанции** J **заблокированы)**

$F(x)$: *все открытые инстанции внутри ссылки x*

Отношение переходов среды

$$\forall(i,j,x)((i,j \in F(x)) \wedge \neg blocked(i,x) \rightarrow < out(i,j) > O(x,i,j) := +1)$$

$$\forall(i,j,x)($$
$$(i,j \in F(x)) \wedge \neg blocked(j,x) \wedge O(x,i,j) > 0 \rightarrow$$
$$< in(i,j) > O(x,i,j) := -1$$
$$)$$

$$\forall(i,j,x)($$
$$(i \in F(x)) \wedge \neg blocked(i,x) \wedge i \notin S(x, wait\ cond\ z, J) \wedge (J \setminus \{i\} \neq \varnothing) \rightarrow$$
$$< cond\ \ z(i,J) > S(x, wait\ cond\ z, J) := \setminus\{i\}$$
$$)$$

Упражнение

Написать остальные правила переходов
Определить функцию blocked
Определить начальное состояние среды