



**САМАРСКИЙ** УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

# Templet Parallel Computing System: Specification, Implementation, Applications

Sergei Vostokin  
Professor, Information Systems and  
Technologies Department,  
Samara University

ITNT-2017, Samara, 25-27 April, 2017



**The Temple system** aims to reduce the parallel programming complexity to the level of sequential programming for ones who develop their own HPC applications

### **Programming in Templet means:**

- a view of parallel algorithm  
*sequential algorithm + specification of parallelism*  
→ *parallel algorithm*
- using standard language (C++), libraries (OpenMP, MPI), IDE, and the Templet *lightweight tooling* in application development

### **Research methods:**

- actor model of execution (Carl Hewitt)
- temporal logic of actions (Leslie Lamport)
- algorithmic skeletons (Murray Cole)
- language-oriented programming (Martin Ward)



# IT IS IMPORTANT TO UNDERSTAND HOW AN APPLICATION INTERACTS WITH ITS EXECUTION ENVIRONMENT

## *Parallel applications*

*Poincaré map*

*Continuous wavelet transform*

*Finite difference method*

*Business process modeling*

*.....  
others*

**Mathematical  
specification  
of computation**

## *Run-time implementations*

*Debugging*

*Sequential execution*

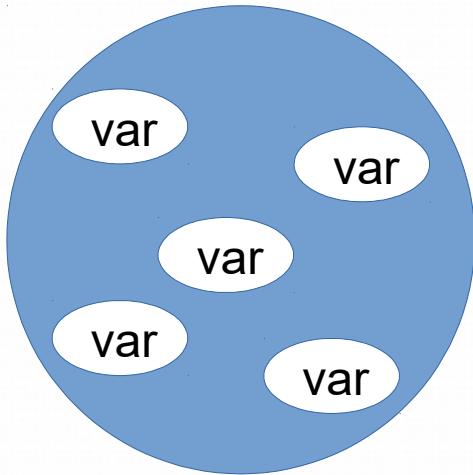
*Parallel execution in shared memory*

*Parallel execution in distributed memory*

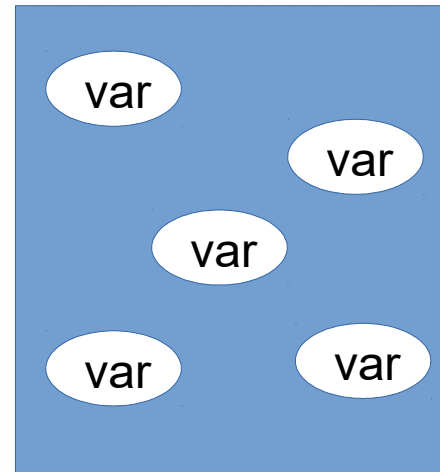
*Emulated execution*



$$F : Var \rightarrow \{actor, message\} \times N$$



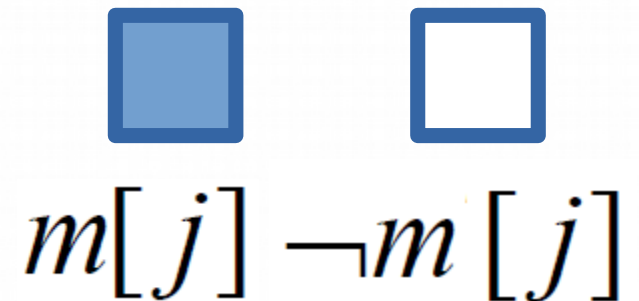
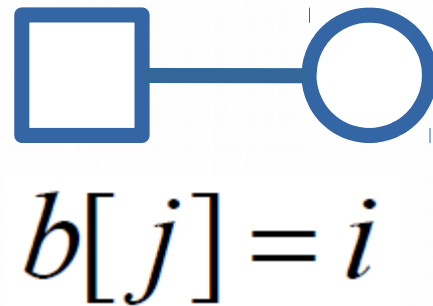
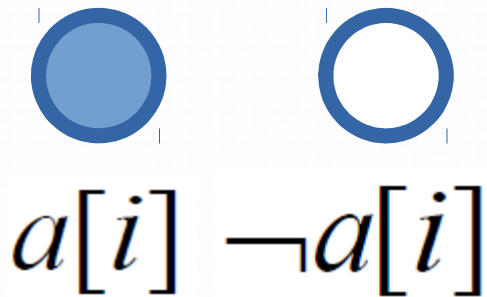
an actor



a message

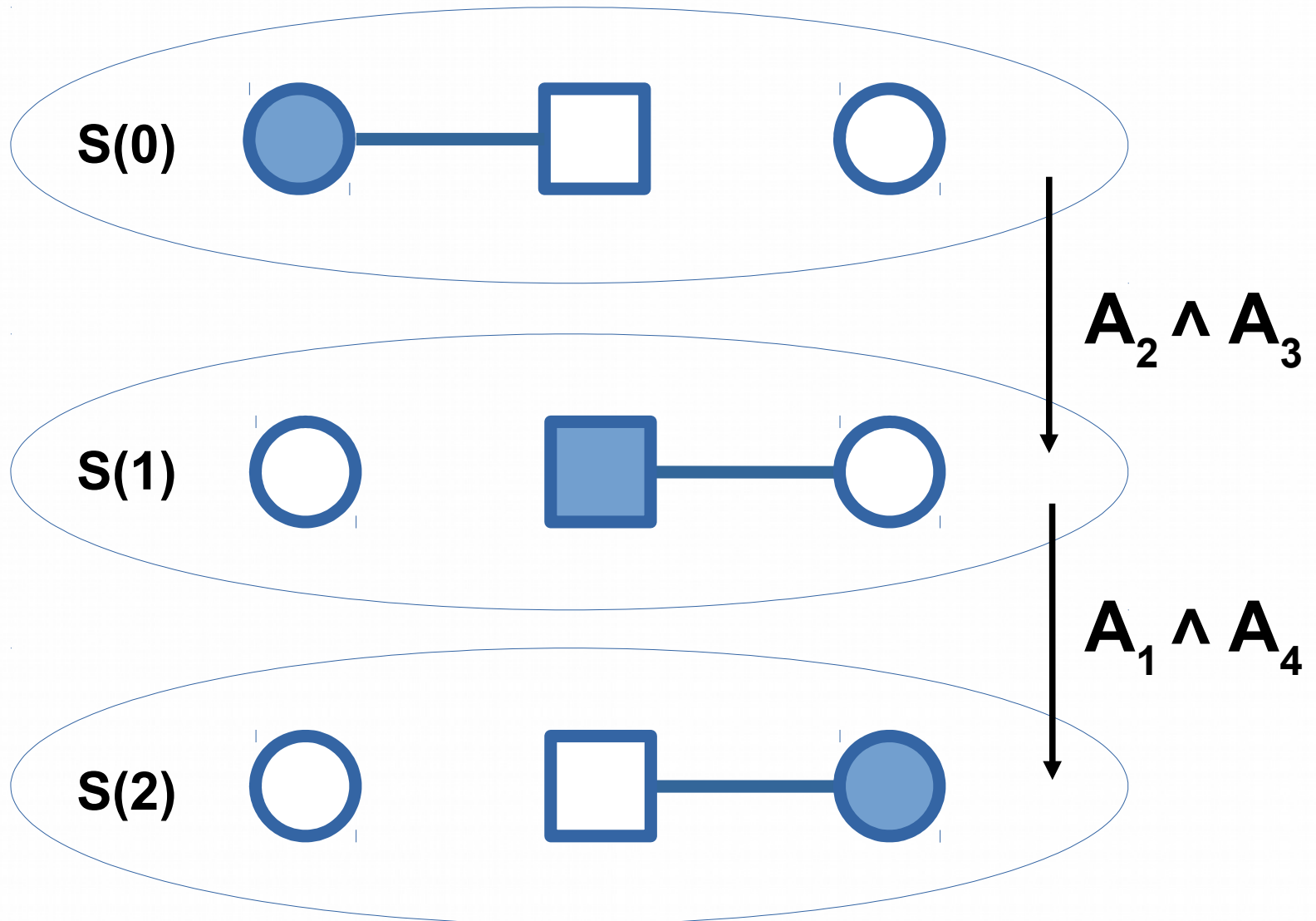


$\{a[1], a[2], \dots, a[i], \dots,$   
 $b[1], b[2], \dots, b[j], \dots,$   
 $m[1], m[2], \dots, m[j], \dots\}$





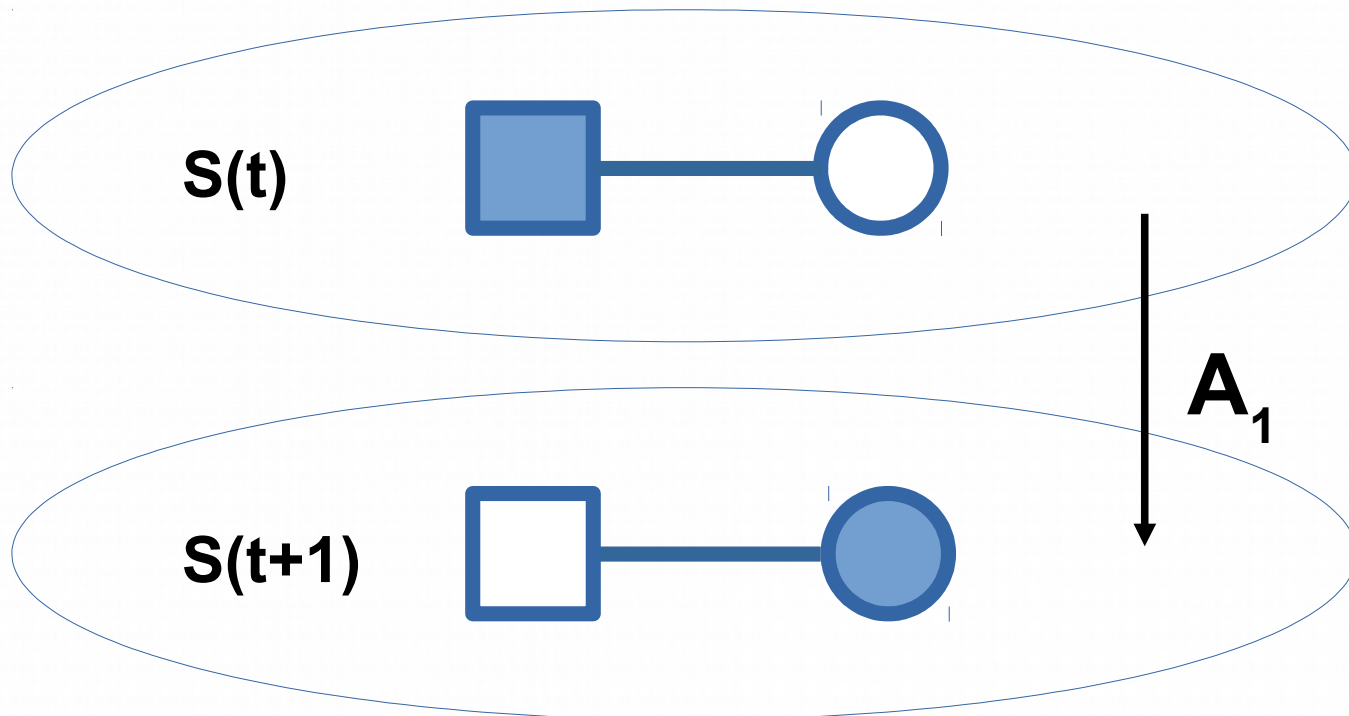
# SEQUENCE OF STATES (ACTIONS) IN THE TRANSMISSION OF A MESSAGE BETWEEN TWO ACTORS





## A1: ACTOR STARTS THE EXECUTION

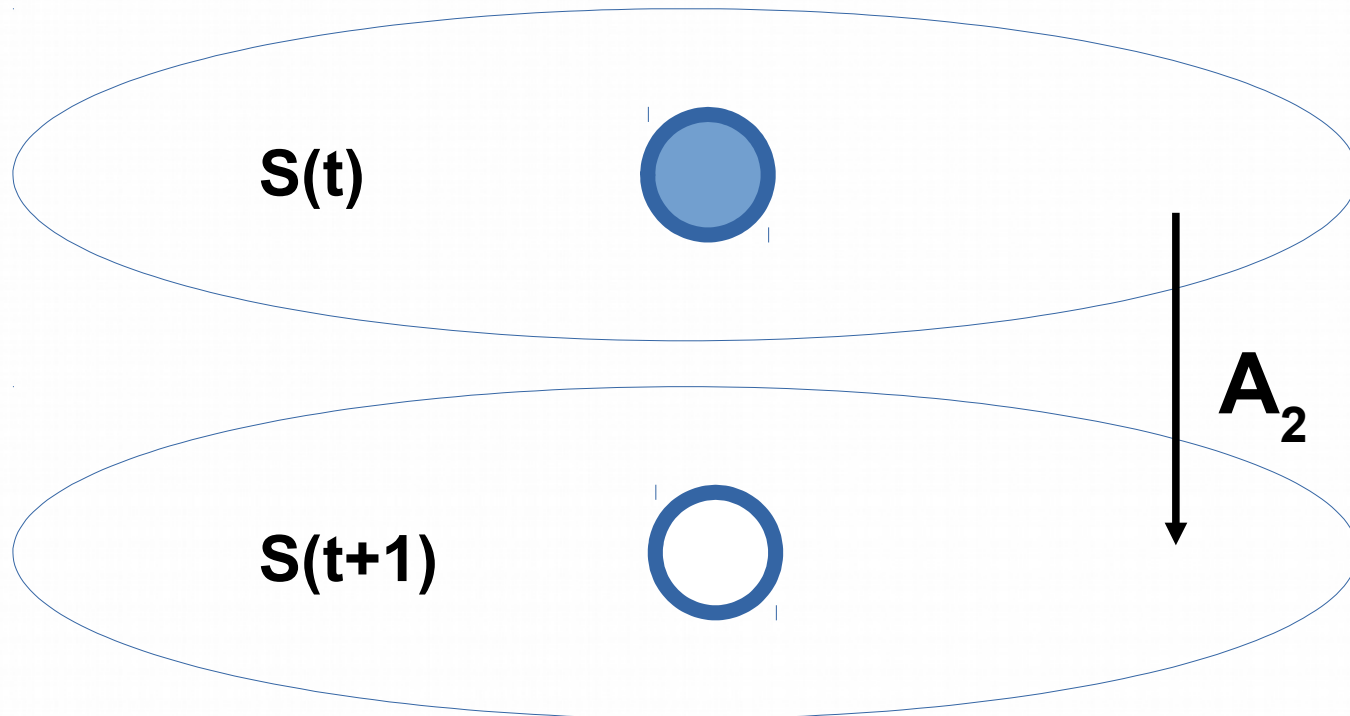
$$A_1 \equiv \exists! j : \neg a[i] \wedge m[j] \wedge b[j] = i \\ \wedge a'[i] \wedge \neg m'[j] \wedge b'[j] = i$$





## A2: ACTOR FINISHES THE EXECUTION

$$A_2 \equiv a[i] \wedge \neg a'[i]$$

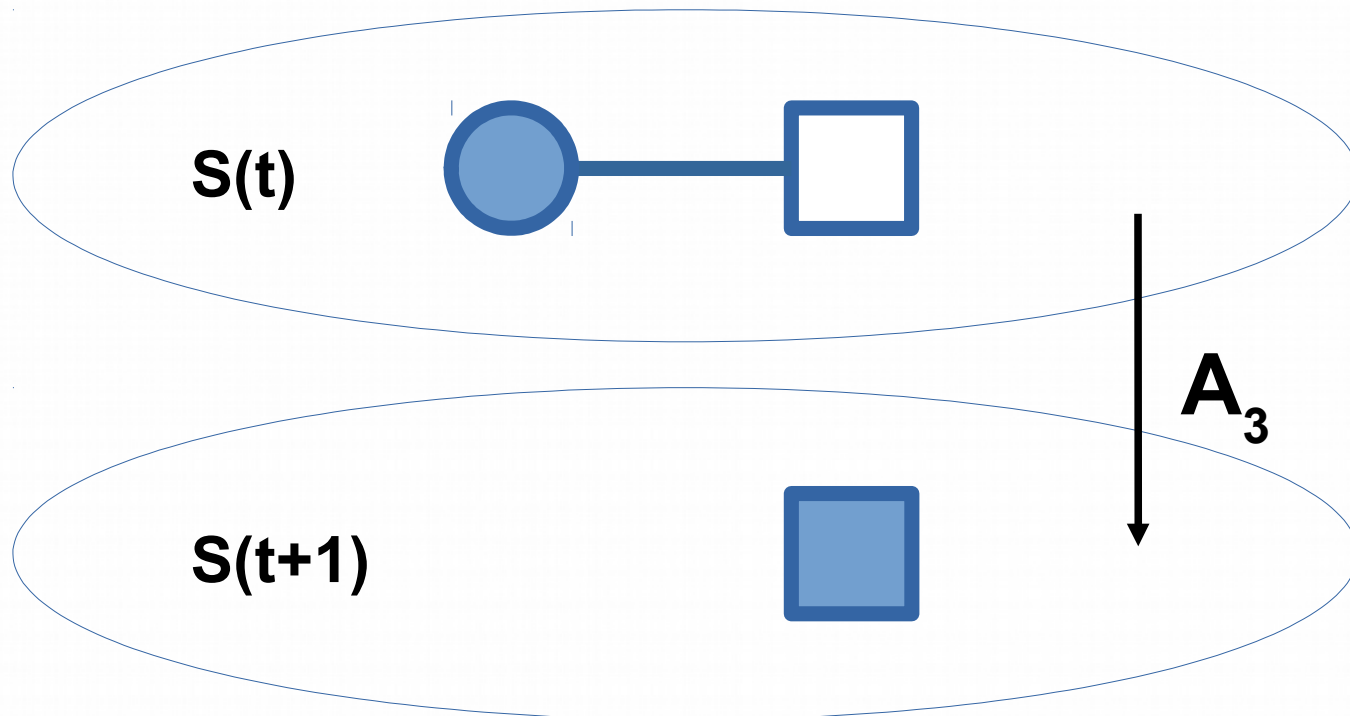






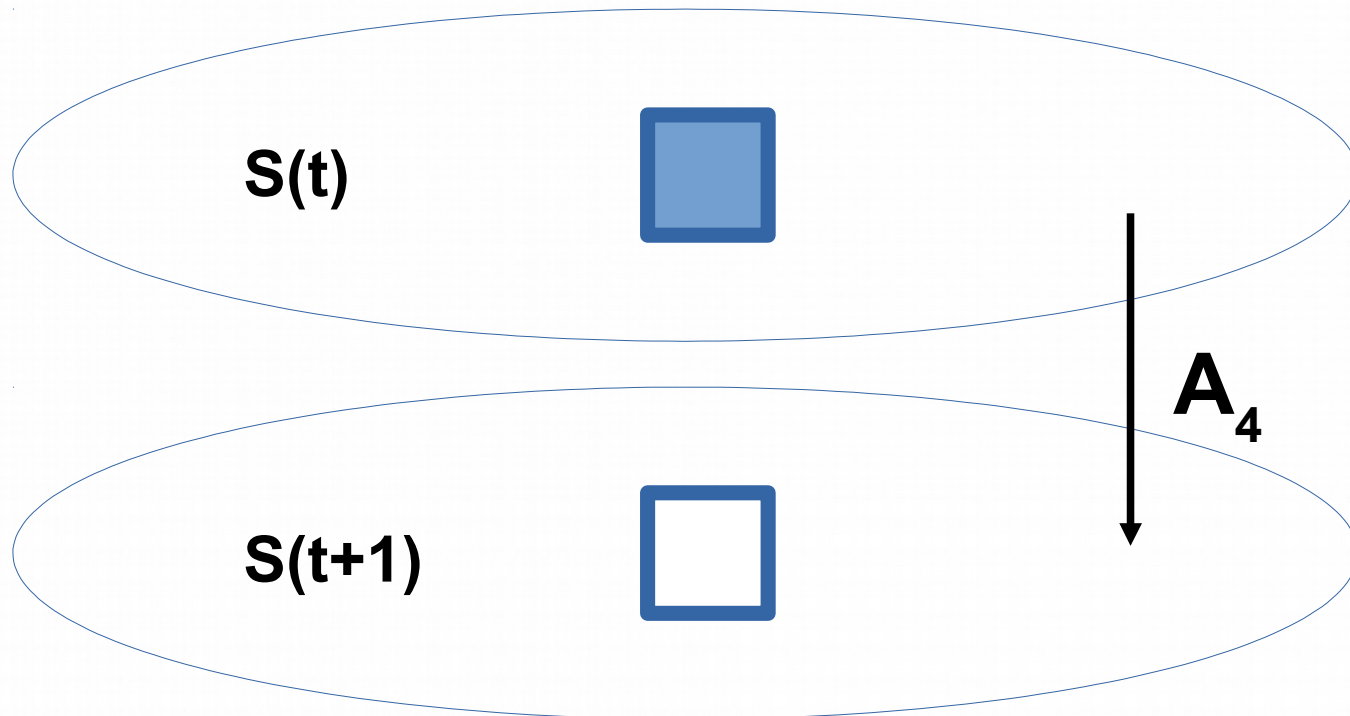
## A3: ACTOR SENDS A MESSAGE

$$A_3 \equiv \exists i : a[i] \wedge b[j] = i \\ \wedge \neg m[j] \wedge m'[j]$$





$$A_4 \equiv m[j] \wedge \neg m'[j]$$





$$I \equiv \exists i : a[i] \vee \exists j : m[j]$$

$$f_1 \equiv a[i] \quad f_2 \equiv (m[j], b[j])$$

$\square [A]_f$  means that the action  $A \vee (f = f')$  executes for every pairs of system states

$WF_f(A)$  means that if the action  $A$  (that change variables  $f$ ) is enabled long enough it will finally be executed



$$\begin{aligned} S \equiv & I \wedge \square [A_1 \vee A_2]_{f_1} \\ & \wedge \square [A_3 \vee A_4]_{f_2} \\ & \wedge WF_{f_1}(A_2) \\ & \wedge WF_{f_2}(A_4) \end{aligned}$$



The message handler procedure *RECV(i,j)*  
is called with the action

$$\begin{aligned} \mathit{recv}_{(call)}(i, j) \equiv & \neg a[i] \wedge m[j] \wedge b[j] = i \\ & \wedge a'[i] \wedge \neg m'[j] \wedge b'[j] = i \end{aligned}$$

and returns with the action

$$\mathit{recv}_{(return)}(i) \equiv a[i] \wedge \neg a'[i]$$

of the actor run-time system **S**



Testing for the accessibility of a message

$$\mathit{access}(i, j) \equiv b[j] = i \wedge \neg m[j]$$

Sending a message to an actor

$$\mathit{send}(i, j) \equiv b'[j] = i \wedge m'[j]$$



- (A) the handler  $RECV(I, \_)$  can access actor state variables  $var$ , if  $F(var)=I$
  
- (B) the handler  $RECV(I, \_)$  can access message state variables  $var$ , if  $F(var)=J$  and  $access(I,J)=true$
  
- (C) the handler  $RECV(I, \_)$  can send a message with the call  $send(\_, J)$ , if  $access(I,J)=true$



```
// engine
struct engine{ std::vector<message*> ready;};
// actor objects
struct actor{ void(*recv) (actor*,message*);};
// message objects
struct message{ actor*a; bool sending;};

inline void send(engine*e, actor*a, message*m) {
    if (m->sending) return;
    m->sending = true;
    m->a = a;
    e->ready.push_back(m);
}
```





```
inline bool access(actor*a, message*m) {
    return m->a == a && !m->sending;
}
inline void run(engine*e) {
    size_t rsize;
    while (rsize = e->ready.size()) {
        int n = rand() % rsize;
        auto it = e->ready.begin() + n;
        message*m = *it; e->ready.erase(it);
        m->sending = false;
        m->a->recv(m->a, m);
    }
}
```



For more details, please, see the poster section presentation:

*S. Vostokin, E. Skoryupina*

A Performance Analysis of Simple Runtime System for Actor Programming in C++



# THE TEMPLLET SYSTEM RUN-TIME AND APPLICATIONS

The screenshot shows a web browser window displaying the GitHub repository page for 'The Templet Markup Language'. The browser's address bar shows the URL 'https://github.com/templet-language'. The page header includes navigation links for 'Features', 'Business', 'Explore', and 'Pricing', along with a search bar and a 'This organization' dropdown. The repository name 'The Templet Markup Language' is prominently displayed, along with its description: 'a tool for concurrent, actor-oriented, skeleton programming'. The location 'Samara, Russia' and the website 'http://templet.ssau.ru' are also visible. Below the repository name, there are tabs for 'Repositories' and 'People'. A search bar for repositories is present, along with a 'Type: All' dropdown. The repository 'newtemplet' is listed as the latest implementation, written in C++ and updated 4 days ago. A small green line graph is visible next to the repository name. On the right side, there are sections for 'Top languages' (showing C++ and Java) and 'People'.



**САМАРСКИЙ** УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

**THANK YOU**

<http://templet.ssau.ru>

<https://Github.com/templet-language>

E-mail: Sergei Vostokin <[easts@mail.ru](mailto:easts@mail.ru)>

34, Moskovskoye shosse, Samara, 443086, Russia  
Tel.: +7 (846) 335-18-26, факс: +7 (846) 335-18-36  
[www.ssau.ru](http://www.ssau.ru), e-mail: [ssau@ssau.ru](mailto:ssau@ssau.ru)