VERIFYING A PROTOCOL ALGEBRAICALLY USING CCS

R. Milner

Rapporteur: Mr. A.M. Koelmans

(Copies of Transparencies)

VERIFYING A PROTOCOL ALGEBRAICALLY USING CCS

In CCS:

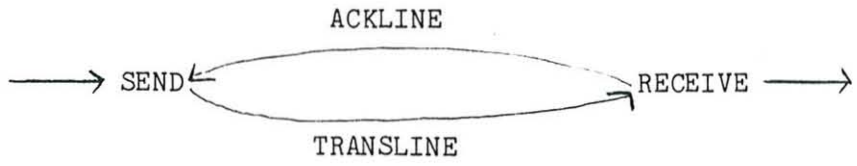
- Algebraic expressions stand for states of machines
- The composition of expressions (= machines) is again an expression (= machine)
- Proofs may be done rigorously by algebraic manipulation

The aim of this talk:

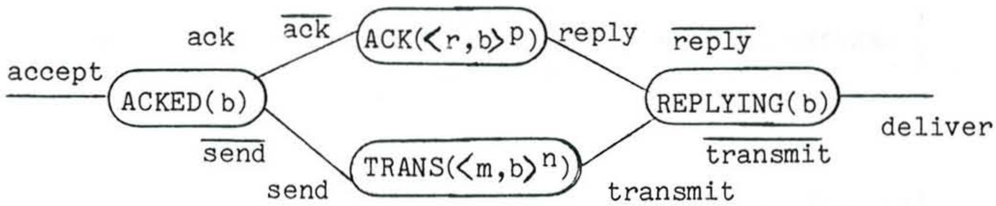
- To illustrate this by a proof of the ALTERNATING-BIT PROTOCOL
- To see the implications for using computer assistance in the proof.

VERIFYING THE ALTERNATING-BIT PROTOCOL, USING BISIMULATION IN CCS

The rough picture



A typical state (with the LINES modelled as agents):



. SENDER HAS RECEIVED ack(b), is ready to accept

. RECEIVER has done $\overline{\text{deliver}}$(m), is performing $\overline{\text{reply}}$(r,b)

. TRANSLINE holds n copies of m paired with bit b

. ACKLINE holds p copies of r paired with bit b

Expressing the System in CCS (ignoring m,r):

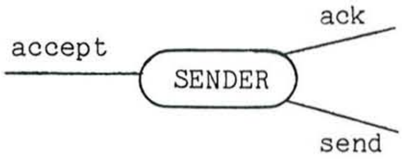
$$\text{SYSTEM}(b,n,p) \triangleq$$

$$\text{ACKED}(b) \,||\, \text{TRANS}(b^n) \,||\, \text{ACK}(b^p) \,||\, \text{REPLYING}(b)$$

- where the four components have yet to be defined.

DEFINING THE SENDER AND RECEIVER

They must cater for the <u>loss</u> or <u>duplication</u> of any message by TRANSLINE or ACKLINE



negation

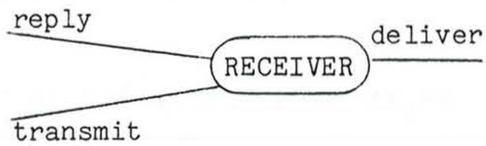
$\mathrm{ACKED}(b) = (\mathrm{ack}(b)+\mathrm{ack}(\tilde{b})).\mathrm{ACKED}\ (b) + \mathrm{accept}.\mathrm{SENDING}(\tilde{b})$	(ignore all ack's)
$\mathrm{SENDING}(b) = (\overline{\mathrm{send}}(b)+\mathrm{ack}(\tilde{b})).\mathrm{SENDING}(b) + \mathrm{ack}(b).\mathrm{ACKED}(b)$	(re-send, ignore re-ack's)

In more convenient notation:

$$\mathrm{ACKED}(b) = (\mathrm{ack}(b)+\mathrm{ack}(\tilde{b}))^{*}.\mathrm{accept}.\mathrm{SENDING}(\tilde{b})$$

$$\mathrm{SENDING}(b) = (\overline{\mathrm{send}}(b)+\mathrm{ack}(\tilde{b}))^{*}.\mathrm{ack}(b).\mathrm{ACKED}(b)$$

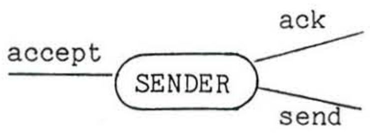
The RECEIVER has a dual definition:

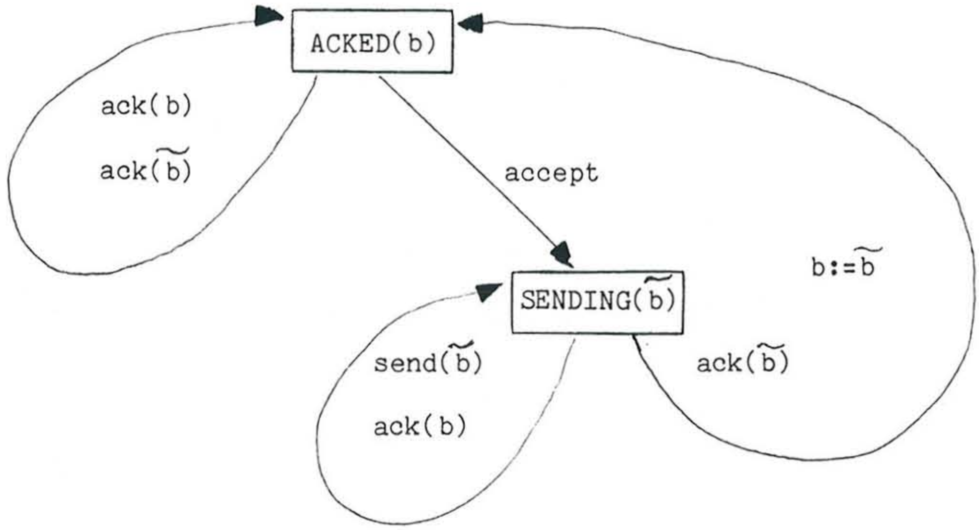


$$\mathrm{TRANSMITTED}(b) = (\mathrm{transmit}(b)+\mathrm{transmit}(\tilde{b}))^{*}.\overline{\mathrm{deliver}}.\mathrm{REPLYING}(b)$$

$$\mathrm{REPLYING}(b) = (\overline{\mathrm{reply}}(b)+\mathrm{transmit}(b))^{*}.\mathrm{transmit}(\tilde{b}).\mathrm{TRANSMITTED}(\tilde{b})$$

STATE DIAGRAM FOR THE SENDER





In state ACKED(b):

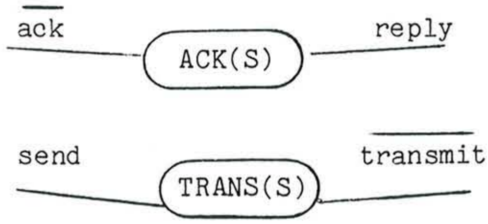
- all acks are ignored

In state SENDING($\tilde{b}$):

- ack(b) is ignored
- re-sending occurs arbitrarily often

DEFINING THE LINES

ACKLINE and TRANSLINE can hold an arbitrary finite sequence S of messages (bits):



Since they can lose or duplicate bits, they can be defined as labelled transition systems, as follows:

$$\mathrm{ACKS}(bs) \xrightarrow{\overline{ack}(b)} \mathrm{ACK}(s) \qquad \mathrm{ACK}(s_1 bs_2) \xrightarrow{T} \mathrm{ACK}(s_1 bs_2)$$

$$\mathrm{ACK}(s) \xrightarrow{reply(b)} \mathrm{ACK}(s) \qquad \mathrm{ACK}(s_1 bs_2) \xrightarrow{T} \mathrm{ACK}(s_1 bbs_2)$$

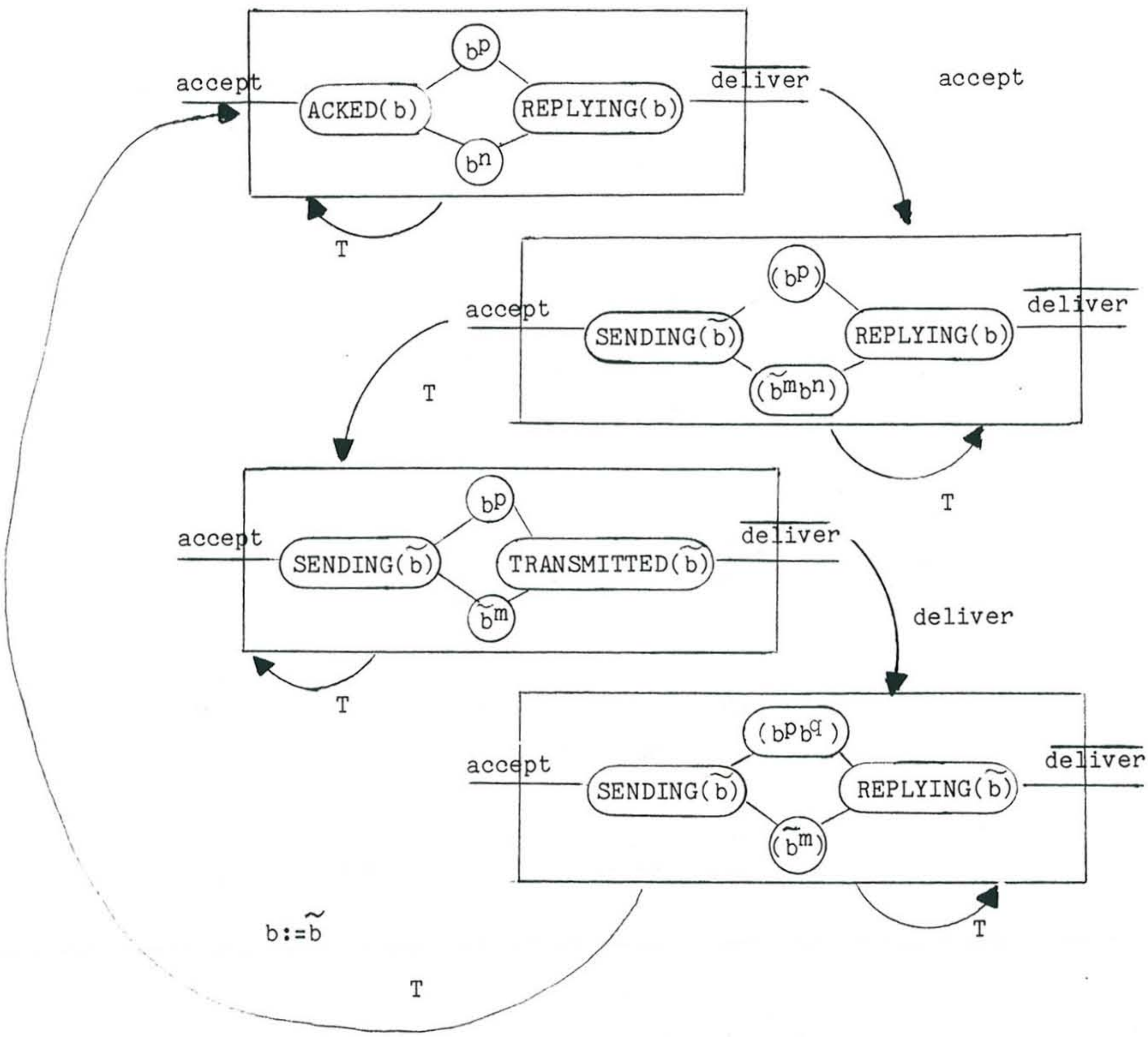
Note: T is an antonomous transition

$$\mathrm{TRANS}(S) \xrightarrow{send(b)} \mathrm{TRANS}(bs) \qquad \mathrm{TRANS}(s_1 bs_2) \xrightarrow{T} \mathrm{TRANS}(s_1 s_2)$$

$$\mathrm{TRANS}(sb) \xrightarrow{\overline{transmit}(b)} \mathrm{TRANS}(s) \qquad \mathrm{TRANS}(s_1 bs_2) \xrightarrow{T} \mathrm{TRANS}(s_1 bbs_2)$$

These agents can be defined also by equations, exactly in the same way as the SENDER and RECEIVER.

THE ACCESSIBLE SYSTEM-STATES



Note:

TRANSLINE,ACKLINE only hold sequences S of the form b^n or $\tilde{b}^m b^n$

The T transitions represent both internal communications and loss or duplication of bits.

THE SYSTEM SPECIFICATION

We wish to prove that the SYSTEM, in initial state

SYSTEM(b,n,p)

is equivalent to a message-buffer of capacity 1.

Thus, the SPECIFICATION is

$$\boxed{\text{SPEC} = \text{accept}.\overline{\text{deliver}}.\text{SPEC}}$$

and we prove

$$\boxed{\text{SYSTEM(b,n,p)} \approx \text{SPEC}}$$

bismulation

The bisimulation can be established mechanically, using mainly the EXPANSION THEOREM of CCS.

[During this development, the ACCESSIBLE SPITES of SYSTEM(b,n,p) are automatically discovered]

The notation of BISIMULATION is due to David Park.

WHAT IS INVOLVED IN THE MECHANICAL PROOF?

Many different formalisms:

- CCS expressions E
- Equations $E_1 \approx E_2$
- Transitions $E_1 \xrightarrow{\mu} E_2$
- Sequence algebra, for sequences $\tilde{b}^m b^n$ etc
- Arithmetic, for the indices m,n
- INFERENCE, involving statements about these things.
- DIAGRAMS!

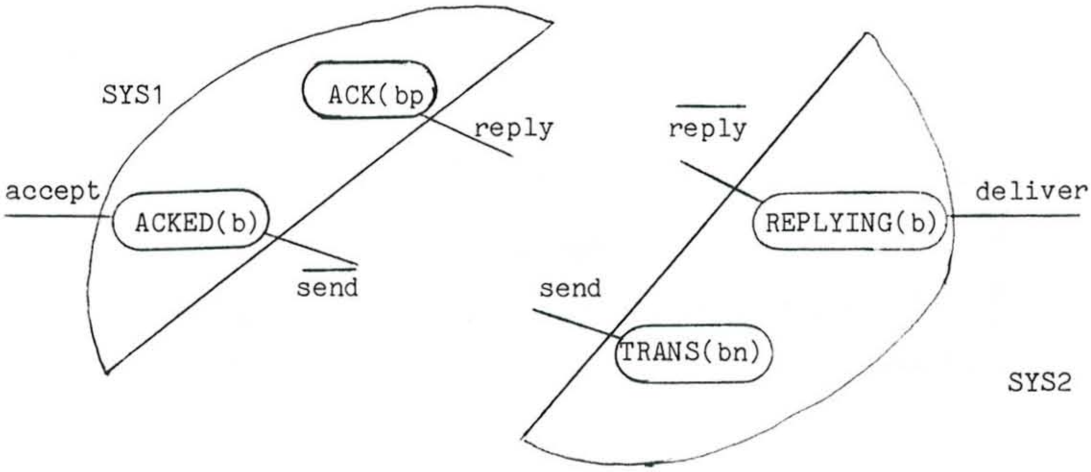
The challenge for machine-assisted proof:

To conduct the verification, with our assistance, using OUR BEST NOTATIONS for all these things, NOT one Procrustian Notation!

ALTERNATIVE PROOF, USING DECOMPOSITION (with Kim Larsen)

Consider the decomposition:

SYSTEM(b,n,p) = SYS1(b,p) $\|$ SYS2(b,n)



Now we are concerned with the behaviour of SYS1(b,p) only in the CONTEXT

$$\boxed{C_2 = [\ ] \| \text{SYS2}(b,n)}$$

We therefore seek

(1) A LANGUAGE (= set of action sequences) L_2 which contains all action sequences permitted by C_2 to any inhabitant.

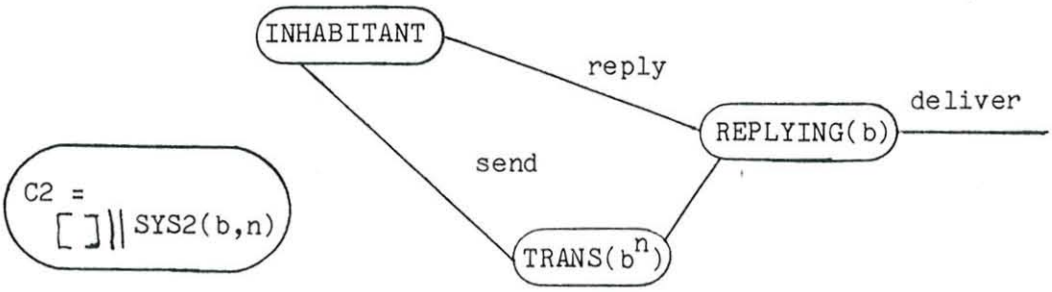
This L_2 is called a Sufficient Inner Environment (SIE) for C_2

(2) A Specification SPEC1(b) such that

$$\boxed{\text{SYS1}(b,p) \approx_{L_2} \text{SPEC1}(b)}$$

This is called bisimulation equivalence <u>relative to L_2</u>.

WHAT DOES C2 PERMIT?:



C2 imposes a constraint on its inhabitant, with respect to $\overline{send}$ and reply actions. Intuitively:

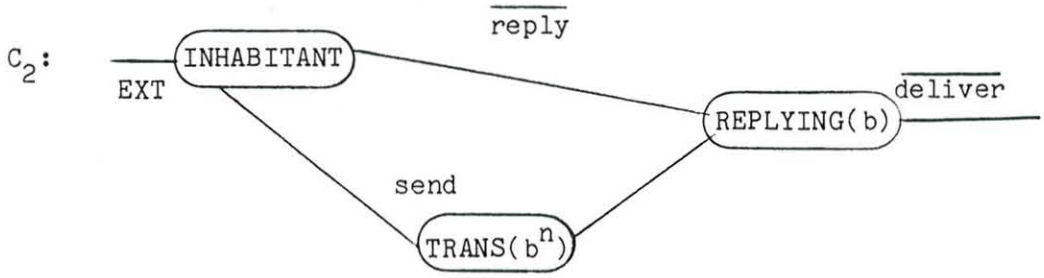
PHASE 1:	All reply actions must be reply(b) until $\overline{send}$(b) occurs;
PHASE 2:	Trivially, all reply actions are reply(b) until reply($\tilde{b}$);
PHASE 3:	Thereafter, provided $\overline{send}$(b) has <u>not</u> occurred during PHASE 2, the whole constraint applies again with b and $\tilde{b}$ interchanged.

The action sequences thus described can be defined as follows, using ACT for all actions and EXT to stand for ACT - $\{send, \overline{send}, reply, \overline{reply}\}$:

$$L_2 = L_2(b) = (EXT+\overline{send}(b)+reply(b))^*.\overline{send}(\tilde{b}).L_2{}'(b)$$

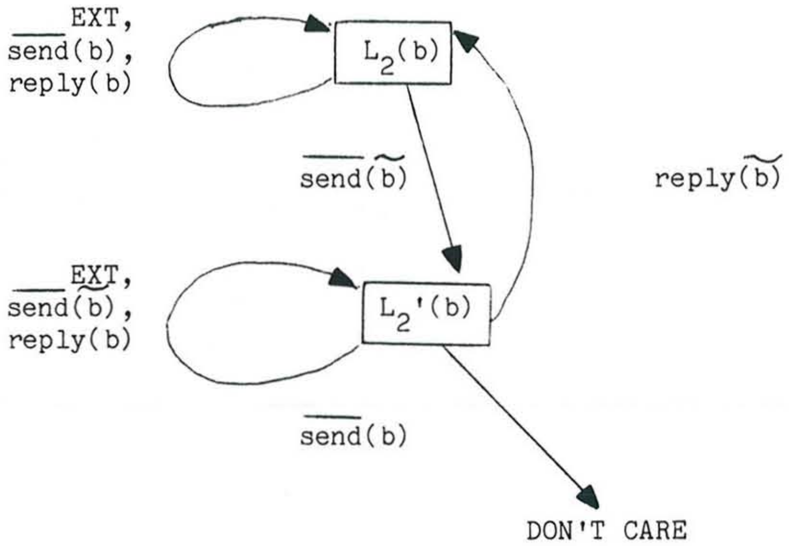
$$L_2{}'(b) = (EXT+\overline{send}(\tilde{b})+reply(b))^*.(\overline{send}(b).ACT + reply(\tilde{b}).L_2(\tilde{b})).$$

A GRAPHICAL PRESENTATION OF L_2, the Sufficient Inner Environment for C_2

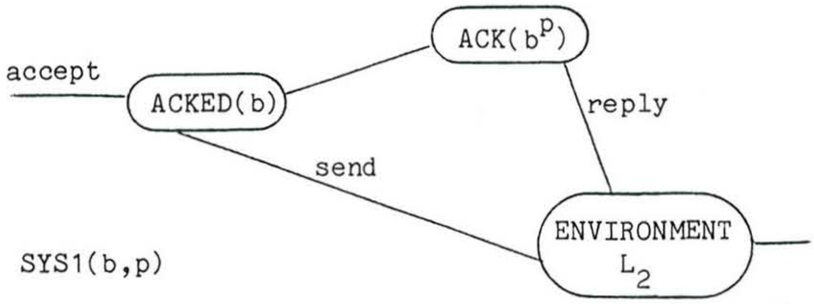


$$L_2(b) = (EXT+\overline{send}(b)+reply(b))^*.\overline{send}(b).L_2'(b)$$

$$L_2'(b) = (EXT+\overline{send}(b)+reply(b))^*.(\overline{send}(b).ACT+reply(\tilde{b}).L_2(\tilde{b}))$$



THE SPECIFICATION OF SYS1 RELATIVE TO THE LANGUAGE L_2



Intuition:

The Environment L_2 ensures that no reply (b) can occur until ACKED(b) has done accept followed by send (b).

Thereafter, SYS1 ensures that no send (b) can occur until the Environment allows reply (b).

Together, it is assured that the ACKLINE can only hold sequences with at most a single bit-change.

We can prove

$$\boxed{\mathrm{SYS1}\ (b,p) \underset{L_2}{\approx} \mathrm{SPEC1}(b)}$$

where SPEC1 has the following definition:

$$\boxed{\begin{array}{rl} \mathrm{SPEC1}(b) = & \mathrm{reply}(b)^*.\mathrm{accept}. \\ & (\overline{\mathrm{send}}(\tilde{b})+\mathrm{reply}(b))^*.\mathrm{reply}(\tilde{b}). \\ & (\mathrm{send}(\tilde{b})+\mathrm{reply}(b)+\mathrm{reply}(\tilde{b}))^*. \\ & \tau.\mathrm{SPEC1}(\tilde{b}) \end{array}}$$

This, unlike SYS1, is a <u>finite-state</u> agent!

CONTINUING THE PROOF

We have shown

$$\boxed{\mathrm{SYSTEM}(b,n,p) \approx \mathrm{SPEC1}(b) \,||\, \mathrm{SYS2}(b,p)}$$

We now have the context C_1 = SPEC1(b) ||[ ] inhabited by SYS2(b,p).

By similar means, we can find an SIE L_1 for C_1; We then find a finite-state SPEC2(b) such that

$$\boxed{\mathrm{SYS2}(b,p) \underset{L_1}{\approx} \mathrm{SPEC2}(b)}$$

Thus, we have shown that

$$\boxed{\mathrm{SYSTEM}(b,n,p) \approx \mathrm{SPEC1}(b) \,||\, \mathrm{SPEC2}(b)}$$

The final step is therefore to prove that

$$\boxed{\mathrm{SPEC1}(b) \,||\, \mathrm{SPEC2}(b) \approx \mathrm{SPEC}}$$

- and this is simple, because SPEC1 and SPEC2 are simple.

NOTE: Some of the material presented by Professor Milner is not included in these proceedings.

DISCUSSION

Dr. Cerf asked how important are the binary nature of your protocols to make CCS work out? Doesn't it get very complicated in bigger cases?

Professor Milner replied, I suspect this is very much the case. We have to try and see what happens.

Professor Tiernary asked, does CCS allow you to prove the same things as the state transition method?

Professor Milner stated, I have proved that the system behaves as a machine that accepts and delivers. There can be no deadlocks, although infinite internal loops may occur. So perhaps I'm not proving as much as I should.

Question: Doesn't one need to add time in order to really express protocols properly?

Professor Milner replied, this is one of the tradeoffs to be made. CCS would be simpler if time was added to it, but then it would be difficult to prove anything. It is an open problem.