Info about homepage & tutorials

- For slides, handouts, typo corrections, and other information, see `http://www.cs.bath.ac.uk/~cf/teaching`
- You can give the tutors your solutions of the exercises and ask for feedback.

Enumerability
& diagonalization

Cantor's Zig-Zag

Pairs of integers: Cantor's Zig-Zag.

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	⋯
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	
⋮					⋱

$f(1) = (1,1), f(2) = (1,2), f(3) = (2,1), f(4) = (1,3), f(5) = (2,2), \ldots$

Exercise

Show the following statements.

1. Every finite set is enumerable.
2. If a **non-empty** set A is enumerable, then it is enumerable by a **total** function. Why does this not work for the empty set?

Code numbers

Definition. Given an enumeration f of a set A, a **code numbers** of an element a of A is a number n such that $f(n) = a$.

Examples:

- The code number of $(1, 3)$ with respect to Cantor's Zig-Zag is 4.
- For the enumeration $2, 2, 4, 4, 6, 6, \ldots$ of the even numbers, the code numbers of 4 are 2 and 3.

Encodings

Definition. An **encoding** of a set A is a total injective function $c : A \longrightarrow N$ into the natural numbers. For a in A, the number $c(a)$ is called the **code** of a.

Proposition. A set A has an encoding if and only if it is enumerable.

The proof of this proposition follows on the next two slides.

From encoding to enumeration

Let $c : A \longrightarrow N$ be an encoding. Then an enumeration $f : N \longrightarrow A$ is given by

$$f(n) = \begin{cases} a & \text{if } n = c(a) \\ \text{undefined} & \text{otherwise} \end{cases}$$

f is well-defined (i.e. there is only one a for every n) because c is injective.

From enumeration to encoding

Let $f : N \to A$ be an enumeration. An encoding $c : A \to N$ is given by

$$c(a) = \text{some } n \text{ such that } f(n) \text{ is equal to } a.$$

Because we need the function c to be an encoding, it must be total and injective. It is total because f is surjective. It is injective because f cannot send some n to two different values a and a'.

“Enumerable” vs. “equinumerous”

Proposition. Every enumerable set A is either finite or equinumerous with N.

Proof. Let A be a set which is enumerable but not finite. Let $c : A \longrightarrow N$ be an encoding of A. We define a bijection $b : N \longrightarrow A$ by

$b(1) =$ the element of A with the smallest code

$b(2) =$ the element of A with the 2nd smallest code

$b(3) =$ the element of A with the 3rd smallest code

$\vdots$

“Enumerable” vs. “equinumerous”

Proposition. A set A is enumerable **if and only if** it is finite or equinumerous with N.

Proof. The left-to-right direction is the result on the previous slide. For the right-to-left direction, suppose first that A is equinumerous with N. Then there is a bijection $b : N \to A$. In particular, A is the range of b, so A is enumerable. Second, suppose that A is finite. Then by an earlier exercise, A is enumerable.

Encoding pairs of integers

As we have seen, Cantor's Zig-Zag provides an encoding of pairs of natural numbers.

Here is an alternative encoding:

$$c_{N\times N}(m,n) = p^m \cdot q^n$$

where p and q are different primes. The total function $c_{N\times N}$ is injective owing to the **uniqueness of prime decomposition**.

Encoding other kinds of pairs

Proposition. If A and B are enumerable sets, then so is the set $A \times B$ of pairs.

Proof. Let $c_A : A \longrightarrow N$ and $c_B : B \longrightarrow N$ be encodings of A and B. An encoding $c_{A \times B} : A \times B \longrightarrow N$ is given by

$$c_{A \times B}(a, b) = c_{N \times N}(c_A(a), c_B(b)),$$

where $c_{N \times N}$ is some encoding of pairs of integers.

($c_{A \times B}$ is injective because c_A, c_B, and $c_{N \times N}$ are.)

Exercises

Show the following statements:

1. If A, B, and C are enumerable sets, then the set of triples

$$A \times B \times C = \{(a, b, c) : a \in A, b \in B, c \in C\}$$

   is enumerable.

2. If $A_1, A_2, \ldots, A_k$ are enumerable sets, then the set of k-tuples $A_1 \times A_2 \times \cdots \times A_k$ is enumerable.

Exercises

Show the following statements:

1. If A is enumerable and there is a surjective function $A \rightarrow B$, then B is enumerable.
2. If B is enumerable and there is a total injective function $A \rightarrow B$, then A is enumerable.

Remark: these two statements are useful for some of the following exercises.

Exercises

Show that the following sets are enumerable:

1. The set Q^+ of positive rational numbers.
2. The set Q of all rational numbers.
3. The set $A \cup B$ for enumerable sets A and B.
4. The set A^* of strings over an enumerable alphabet A.

Exercises

Show that the following statements:

1. The set $P_{fin}(N)$ of **finite subsets** of N is enumerable.
2. The set $P_{fin}(A)$ of finite subsets of an arbitrary enumerable set A is enumerable.

The limits of enumerability

- So far, we have seen various examples of enumerable sets.
- Next, we shall see counterexamples.
- This is important, because only enumerable sets can be used in computations.

The set of sets of natural numbers

Theorem.[Cantor's Theorem] The set $P(N)$ (powerset of the natural numbers) is not enumerable.

Cantor's diagonal argument

The "diagonal argument" is Cantor's celebrated proof of his theorem. Here is the idea:

The proof proceeds **by contradiction**—that is, we assume that $P(N)$ is enumerable and show that this leads to a contradiction.

So suppose that $P(N)$ is enumerable. Then it has an enumeration $s : N \to P(N)$. To obtain the contradiction, we define a set $\Delta(s)$ of natural numbers which cannot be in the range of s.

Cantor's diagonal argument

The set $\Delta(s)$ is defined as follows: for each pos. integer n,

$$n \in \Delta(s) \text{ if and only if } n \notin s(n)$$

(Note the similarity wit Russell's Paradox!) To show that $\Delta(s)$ is not in the range of s, we use again a proof by contradiction. So suppose that $\Delta(s)$ **is** in the range of s, that is, $\Delta(s) = s(m)$ for some m. Then

$$m \in \Delta(s) \text{ if and only if } m \in s(m)$$

But this is a contradiction to the definition of Δ.

q.e.d.

Idea behind the diagonal argument

Think of the set $s(i)$ of natural numbers as a function $s_i : N \longrightarrow \{0, 1\}$ such that

$$s_i(n) = \begin{cases} 1 & \text{if } n \in s(i) \\ 0 & \text{otherwise} \end{cases}$$

(s_i is called the **characteristic function** of the set $s(i)$.)

Idea behind the diagonal argument

For example, if s_i is like in the table below

$$\begin{array}{c|ccccccc} n & 1 & 2 & 3 & 4 & 5 & 6 & \dots \\ \hline s_i(n) & 0 & 1 & 0 & 1 & 0 & 1 & \dots \end{array}$$

it represents the set of even numbers.

Idea behind the diagonal argument

$$
\begin{array}{c|cccccc}
n & 1 & 2 & 3 & 4 & 5 & \dots \\
\hline
s_1(n) & & 0 & 0 & 1 & 0 & \dots \\
s_2(n) & 0 & & 0 & 1 & 0 & \\
s_3(n) & 1 & 0 & & 1 & 0 & \\
s_4(n) & 1 & 0 & 0 & & 1 & \\
s_5(n) & 0 & 0 & 1 & 1 & & \\
\vdots & \vdots & & & & &
\end{array}
$$

Non-enumerability of functions $N \to N$

A similar argument shows that the functions from natural numbers to natural numbers are not enumerable:

Non-enumerability of functions $N \to N$

Exercise

Show the following statements:

1. The set $P(A)$ of all subsets of an infinite enumerable set is non-enumerable.
2. Suppose that we have a programming language, such that every program describes a function $N \to N$. Show that there must be functions $N \to N$ that are described by no program.

Automata

Automata in computer science

In computer science: automaton = abstract computing "machine"

Automata in this lecture

- **Turing machines** (1937) and **abacus machines** (1960s): have all capabilities of today's computers. Used to study the boundary between **computable** and **uncomputable**.
- **Finite automata** (also called **finite state machines**, emerged during the 1940's and 1950's): useful e.g. text search, protocol verification, compilers, descriptions of certain **formal grammars** (N. Chomsky, 1950's).

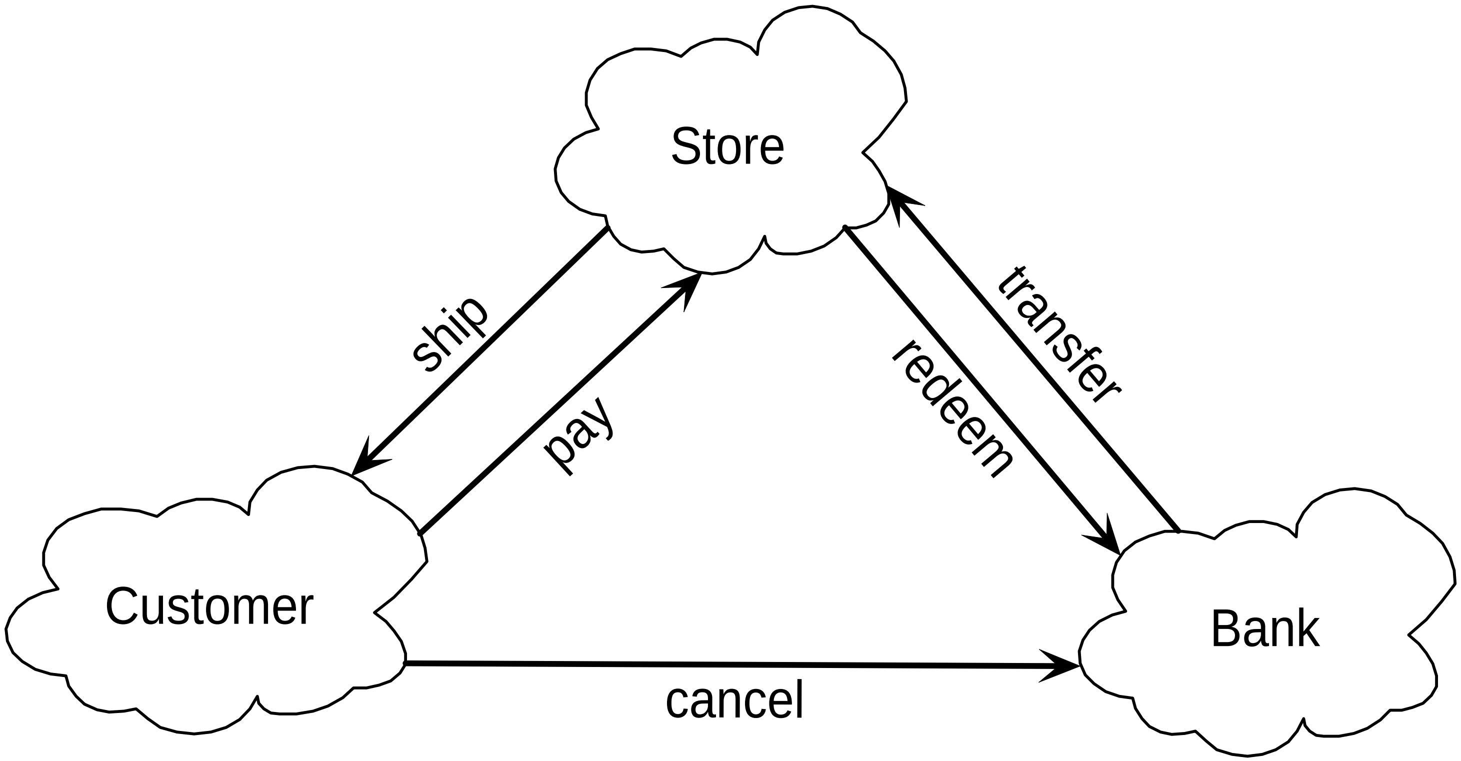
Finite automata

We shall study finite automata first, because they can be seen as a first step towards Turing machines and abacus machines.

Uses of finite automata

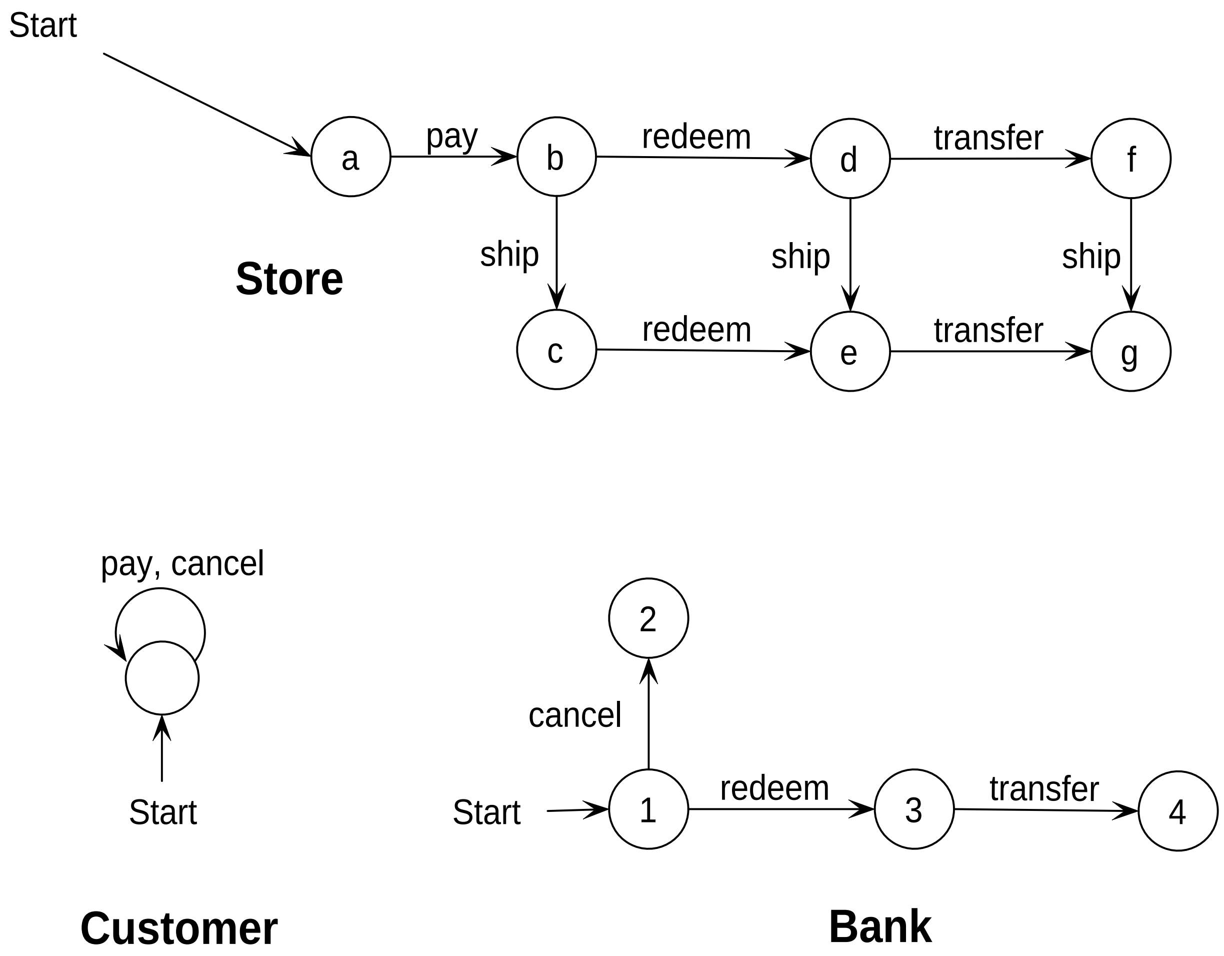
- Used in software for verifying all kinds of systems with a finite number of states, such as communication protocols
- Used in software for scanning text, to find certain patterns
- Used in “Lexical analyzers” of compilers (to turn program text into “tokens”, e.g. identifiers, keywords, brackets, punctuation)
- Part of Turing machines and abacus machines

Example: comm. protocol



Customer, Store, and Bank will be finite automata.

Communication protocol



Simulating the whole system

- Idea: running Customer, Store, and Bank “in parallel”.
- Initially, each automaton is in its start position.
- The system can move on for every action that is possible in **each** of the three automata.

The missing irrelevant actions

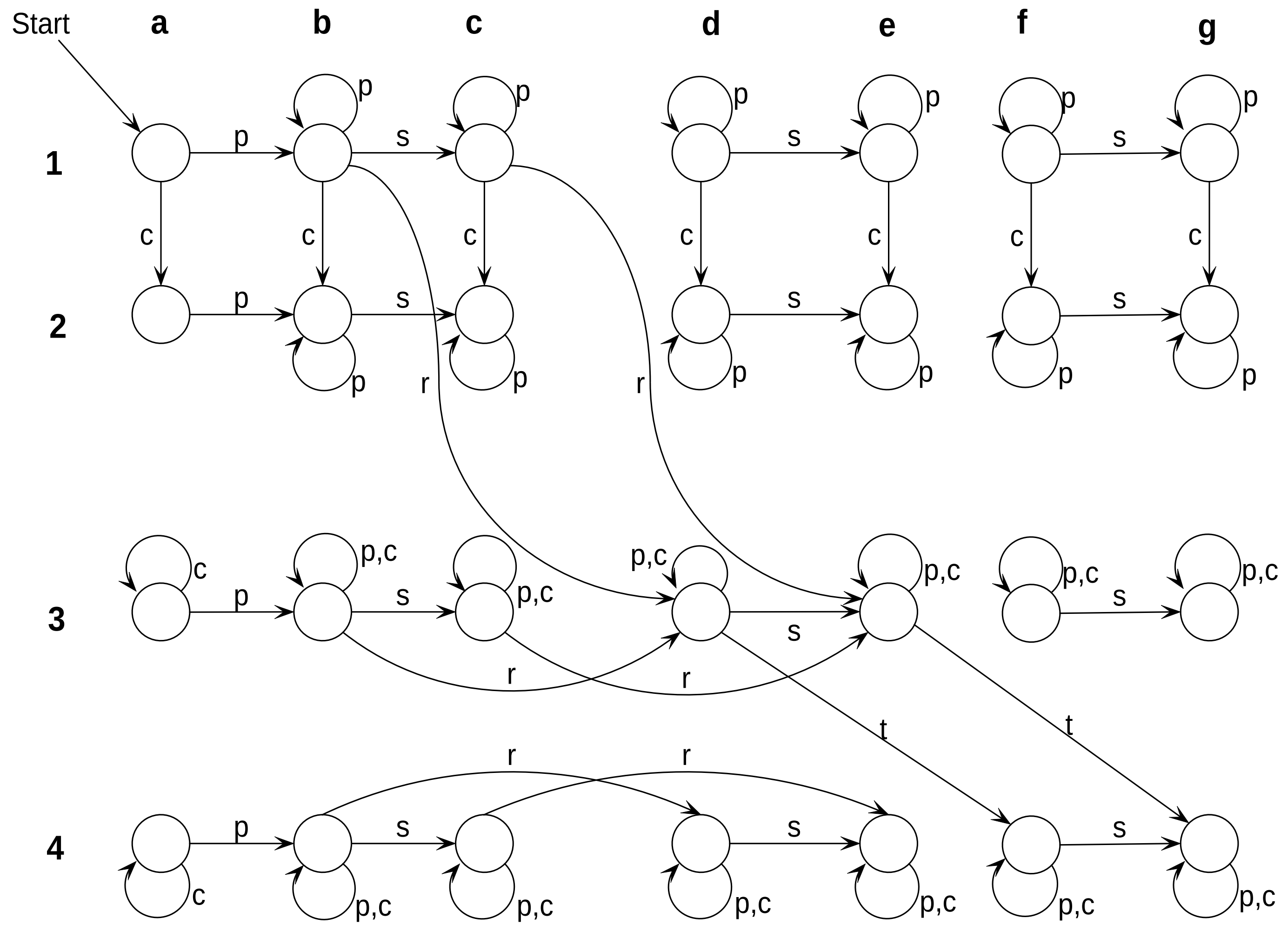
- Problem: Bank gets stuck during the pay action, although paying is only between Customer and Store.
- Solution: we need to add a loop labeled “pay” to state 1 of Bank.
- More generally, we need loops for all such irrelevant actions.
- But illegal actions should remain impossible. E.g. Bank should not allow “redeem” after “cancel”.

Adding irrelevant actions

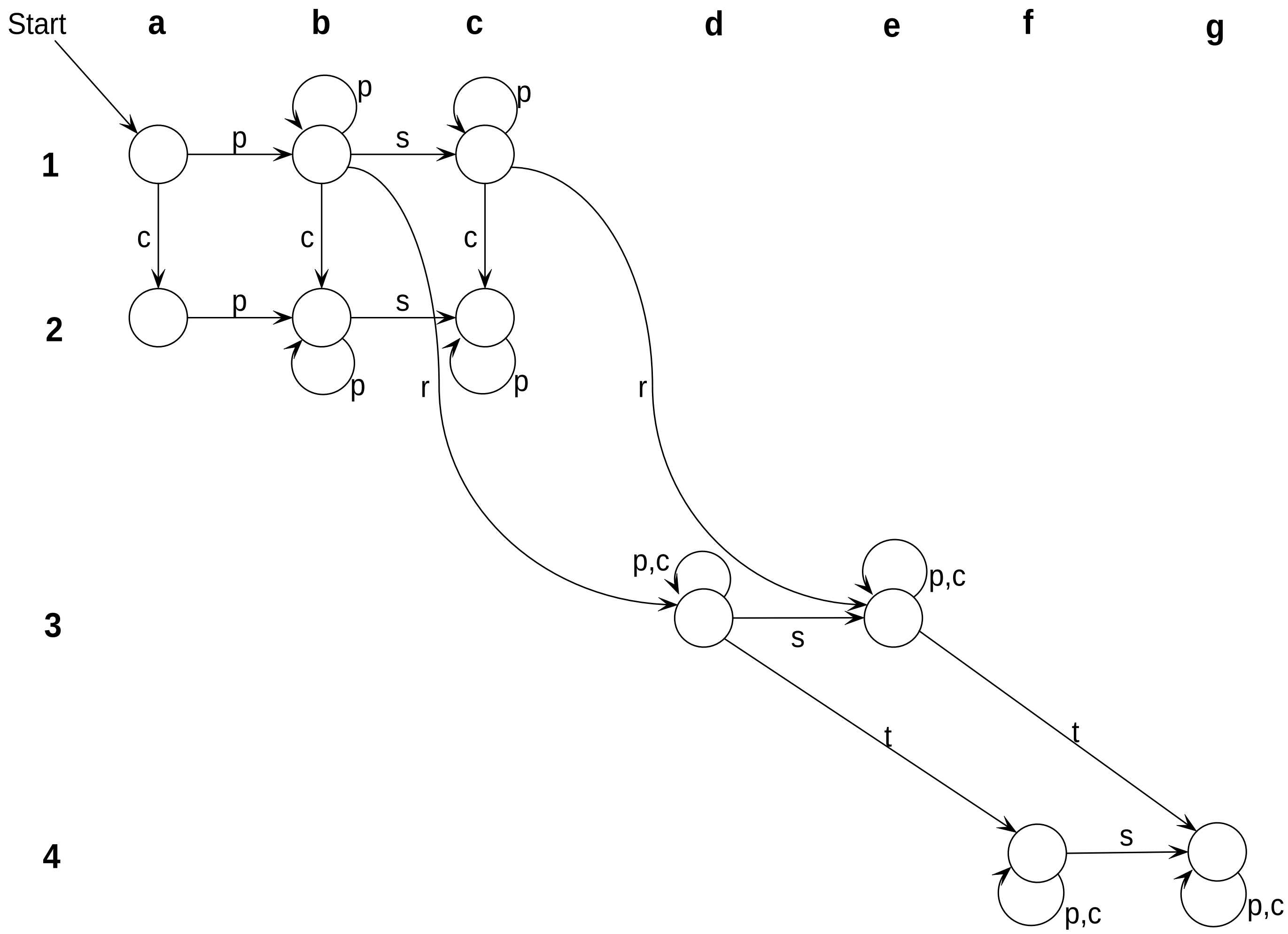
Simulating the whole system

- Simulation by **product automaton**.
- Its states are pairs $(StoreState, BankState)$, e.g. (a,1) or (c,3). (Because Customer has only one state and allows every action, it can be neglected.)
- It has a transition $(StoreState, BankState) \xrightarrow{action} (StoreState', BankState')$ whenever Store has a transition $StoreState \xrightarrow{action} StoreState'$ and Bank has a transition $BankState \xrightarrow{action} BankState'$.

Product automaton



Without unreachable states



Usefulness for protocol verification

- We can now answer all kinds of interesting questions, e.g. “Can it happen that Store ships the product and never receives the money transfer?”
- Yes! If Customer has indicated to pay, but sent a cancellation message to the Bank, we are in state (b,2). If Store ships then, we make a transition into (c,2), and the Store will never receive a money transfer!
- So store should never ship before redeeming.