

# CIS 505 Software Systems

## Lecture Note on CSP

Instructor: Insup Lee  
 Department of Computer and Information Science  
 University of Pennsylvania

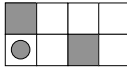
[The slides are originally prepared by U. Sammapun,  
 based on the CSP book by C.A.R. Hoare]

## Communicating Sequential Processes (CSP)

- Capture real world **behaviors**
  - Chocolate vending machine
- **Event** (instantaneous actions)
  - *coin*
  - *choc*
- **Process** (behavior patterns of objects)
  - *VMC* – chocolate vending machine
- **Alphabet** (set of possible events, denoted as  $\alpha P$ )
  - $\alpha VMC = \{coin, choc\}$
- $STOP_A$  (behavior or process of a broken object with alphabet A)
  - $STOP_{\alpha VMC}$

## Prefix

- **Prefix:**  $(x \rightarrow P)$ 
  - Process with same alphabet as  $P$
  - $\alpha(x \rightarrow P) = \alpha P$  where  $x \in \alpha P$
- Broken chocolate vending machine
  - $coin \rightarrow STOP_{\alpha VMC}$
  - $(coin \rightarrow (choc \rightarrow (coin \rightarrow (choc \rightarrow STOP_{\alpha VMC}))))$
- Counter board
 



  - $\alpha CRT = \{up, right\}$
  - $CRT = (right \rightarrow up \rightarrow right \rightarrow right \rightarrow STOP_{\alpha CRT})$
- **Incorrect**
  - $P \rightarrow Q$
  - $x \rightarrow y$  (Correct syntax:  $x \rightarrow (y \rightarrow STOP)$ )

## Recursion

- Describe **repetitive** behaviors
- Clock
  - $\alpha CLOCK = \{tick\}$
  - $CLOCK = tick \rightarrow CLOCK$
  - $CLOCK$ 
    - =  $(tick \rightarrow CLOCK)$  [original equation]
    - =  $(tick \rightarrow (tick \rightarrow CLOCK))$  [by substitution]
    - =  $(tick \rightarrow (tick \rightarrow (tick \rightarrow CLOCK)))$  [similarly]
    - = .....
    - =  $tick \rightarrow tick \rightarrow tick \rightarrow tick \rightarrow tick \rightarrow \dots$  [unfolding CLOCK many times]
- Useless equation?
  - $X = X$

## Recursion

- **Guarded**
  - A process description that begins with a prefix
  - $CLOCK = (tick \rightarrow CLOCK)$
- If  $F(X)$  is a guarded expression containing process  $X$  and alphabet  $A$ 
  - Then  $X = F(X)$  has a **unique solution** with alphabet  $A$
  - Solution:  $\mu X: A \bullet F(X)$
- $X$  is a local name and can be changed
  - $\mu X: A \bullet F(X) = \mu Y: A \bullet F(Y)$
  - Because a solution for  $X = F(X)$  is also a solution for  $Y = F(Y)$

CIS 505, Spring 2007

CSP

5

## Recursion Examples

- Clock
  - $CLOCK = \mu X: \{tick\} \bullet (tick \rightarrow X)$
- Working chocolate vending machine!
  - $VMC = (coin \rightarrow (choc \rightarrow VMC))$
  - Or formally,  $VMC = \mu X: \{coin, choc\} \bullet (coin \rightarrow (choc \rightarrow X))$
- Machine gives change for 5p repeatedly
  - $\alpha CH5A = \{in5p, out2p, out1p\}$
  - $CH5A = (in5p \rightarrow out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5A)$
- Different change-giving machine with same alphabet
  - $CH5B = (in5p \rightarrow out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5B)$

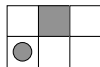
CIS 505, Spring 2007

CSP

6

## Choice

- **Interaction with environment**
  - Machine with 1p coin slot and 2p coin slot
  - It's a customer's choice
- **Syntax**
  - $(x \rightarrow P \mid y \rightarrow Q)$
  - where  $\alpha(x \rightarrow P \mid y \rightarrow Q) = \alpha P$
  - provided  $\{x, y\} \subseteq \alpha P$  and  $\alpha P = \alpha Q$
- **Examples**
  - Possible counter moves
    - $(up \rightarrow STOP \mid right \rightarrow right \rightarrow up \rightarrow STOP)$
  - Machine with two combination of changes
    - $CH5C = in5p \rightarrow (out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5C$
    - $out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5C)$



CIS 505, Spring 2007

CSP

7

## Choice Example

- Chocolate / toffee machine
  - $VMCT = \mu X \bullet coin \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)$
- Machine with choices of coins / goods / change
  - $VM = (in2p \rightarrow (large \rightarrow VM$
  - $\mid small \rightarrow out1p \rightarrow VM)$
  - $\mid in1p \rightarrow (small \rightarrow VM$
  - $\mid in1p \rightarrow (large \rightarrow VM$
  - $\mid in1p \rightarrow STOP)))$
- Machine that trusts customers
  - $VMCRED = \mu X \bullet (coin \rightarrow choc \rightarrow X \mid choc \rightarrow coin \rightarrow X)$
- To prevent loss, an initial payment is required
  - $VM2 = (coin \rightarrow VMCRED)$
- Copying Process
  - $COPYBIT = \mu X \bullet (in.0 \rightarrow out.0 \rightarrow X \mid in.1 \rightarrow out.1 \rightarrow X)$

CIS 505, Spring 2007

CSP

8

## Choice

- **More than two alternatives**
  - $(x \rightarrow P \mid y \rightarrow Q \mid \dots \mid z \rightarrow R)$
  - where  $x, y, z$  are distinct events
- **Incorrect syntax**
  - $(x \rightarrow P \mid x \rightarrow Q)$
  - $(x \rightarrow P \mid (y \rightarrow Q \mid z \rightarrow R))$
- **In general**, choice is written as
  - $(x : B \rightarrow P(x))$
  - Offers a choice of any event  $x$  in  $B$ , and behaves like  $P(x)$
  - A set  $B$  is called the "initial menu" of the process
- **Example**: Process which can engage in any event in alphabet  $A$ 
  - $\alpha RUN_A = A$
  - $RUN_A = (x : A \rightarrow RUN_A)$
- There are other kinds of choice: e.g., *non-deterministic* choice

CIS 505, Spring 2007

CSP

9

## Mutual Recursion

- **Orange-Lemon drink dispenser (DD)**
  - Pressing two buttons: *setorange, setlemon*
  - Dispensing drinks: *orange, lemon*
  - $\alpha DD = \alpha O = \alpha L = \{\text{setorange, setlemon, orange, lemon}\}$

$$DD = (\text{setorange} \rightarrow O \mid \text{setlemon} \rightarrow L)$$

$$O = (\text{orange} \rightarrow O \mid \text{setlemon} \rightarrow L \mid \text{setorange} \rightarrow O)$$

$$L = (\text{lemon} \rightarrow L \mid \text{setorange} \rightarrow O \mid \text{setlemon} \rightarrow L)$$

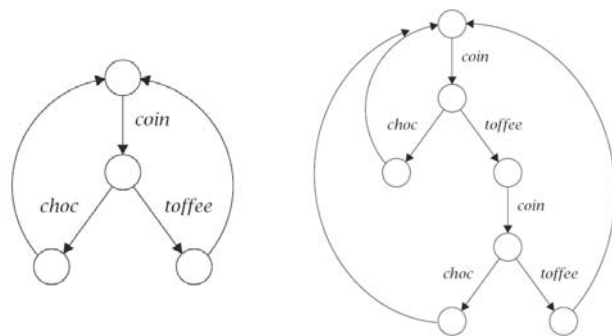
- **Object movement**
  - On the ground: move *up* or *around*
    - $CT_0 = (\text{up} \rightarrow CT_1 \mid \text{around} \rightarrow CT_0)$
  - In the air: move *up* or *down*
    - $CT_{n+1} = (\text{up} \rightarrow CT_{n+2} \mid \text{down} \rightarrow CT_n)$

CIS 505, Spring 2007

CSP

10

## Pictures



CIS 505, Spring 2007

CSP

11

## Laws

- **Law1:**
  - $(x \rightarrow P \mid y \rightarrow Q) = (y \rightarrow Q \mid x \rightarrow P)$
  - $(x \rightarrow P) \neq STOP$
  - $(x \rightarrow P) \neq (y \rightarrow Q)$  if  $x \neq y$
  - $(x \rightarrow P) = (x \rightarrow Q)$  implies  $P = Q$
- Example:
  - $(\text{coin} \rightarrow \text{choc} \rightarrow \text{coin} \rightarrow \text{choc} \rightarrow STOP) \neq (\text{coin} \rightarrow STOP)$
  - $\mu X \cdot (\text{coin} \rightarrow (\text{choc} \rightarrow X \mid \text{toffee} \rightarrow X))$   
 $= \mu X \cdot (\text{coin} \rightarrow (\text{toffee} \rightarrow X \mid \text{choc} \rightarrow X))$

CIS 505, Spring 2007

CSP

12

## Laws

- **Law2:** let  $F(X)$  be a guarded expression
  - $(Y = F(Y)) \equiv (Y = \mu X \cdot F(X))$
  - $\mu X \cdot F(X) = F(\mu X \cdot F(X))$
- Example:
  - Let  $VM1 = (\text{coin} \rightarrow VM2)$   
 $VM2 = (\text{choc} \rightarrow VM1)$
  - Then  $VM1 = (\text{coin} \rightarrow VM2)$   
 $= (\text{coin} \rightarrow (\text{choc} \rightarrow VM1))$   
 $= VMC$

CIS 505, Spring 2007

CSP

13

## Traces

- **Trace of a behavior of a process**
  - **Finite sequence** of symbols recording events up to some moment in time
- **Example**
  - Chocolate vending machine after serving 2 customers
    - $\langle \text{coin}, \text{choc}, \text{coin}, \text{choc} \rangle$
  - Before anyone puts coins in
    - $\langle \rangle$  (called empty trace, the shortest possible trace)
  - Change-giving machine – customer is waiting for the last 2p
    - $\langle \text{in5p}, \text{out2p}, \text{in1p} \rangle$

CIS 505, Spring 2007

CSP

14

## Operations on Traces

- **Concatenation:**  $s \wedge t$ 
  - $\langle \text{coin}, \text{choc} \rangle \wedge \langle \text{coin}, \text{toffee} \rangle = \langle \text{coin}, \text{choc}, \text{coin}, \text{toffee} \rangle$
- **Head:**  $s_0$     **Tail:**  $s'$ 
  - $\langle x, y, x \rangle_0 = x$
  - $\langle x, y, x \rangle' = \langle y, x \rangle$
- **Ordering:**  $s \leq t = (\exists u \cdot s \wedge u = t)$ 
  - "s is a prefix of t"
  - $\langle x, y \rangle \leq \langle x, y, x, w \rangle$
- **Star:**
  - $A^* = \{ t \mid t = \langle \rangle \text{ or } (t_0 \in A \text{ and } t' \in A^*) \}$

CIS 505, Spring 2007

CSP

15

## Operation on Traces

- **After:**  $P / s$  where  $s \in \text{traces}(P)$ 
  - "P after s"
  - $(VMC / \langle \text{coin} \rangle) = (\text{choc} \rightarrow VMC)$
  - $(VMC / \langle \text{coin}, \text{choc} \rangle) = VMC$
- **Restriction:**  $(t \uparrow A)$  "trace t when restricted to symbols in A"
  - $\langle \text{around}, \text{up}, \text{down}, \text{around} \rangle \uparrow \{ \text{up}, \text{down} \} = \langle \text{up}, \text{down} \rangle$
- **Length:**  $\# t$ 
  - $\# \langle x, y, x \rangle = 3$

CIS 505, Spring 2007

CSP

16

## Traces of Process

- **Complete set of all possible traces of  $P$** 
  - $traces(P)$
- **Example**
  - $traces(STOP) = \{ \langle \rangle \}$
  - $traces(coin \rightarrow STOP) = \{ \langle \rangle, \langle coin \rangle \}$
  - $traces(\mu X \bullet tick \rightarrow X) = \{ \langle \rangle, \langle tick \rangle, \langle tick, tick \rangle, \dots \}$   
 $= \{ tick \}^*$
  - **Chocolate vending machine**
    - $trace(\mu X \bullet coin \rightarrow choc \rightarrow X) = \{ s \mid \exists n \bullet s \leq \langle coin, choc \rangle^n \}$

CIS 505, Spring 2007

CSP

17

## Specifications

- **Describe intended behaviors of products**
  - Assume  $tr$  is a variable for an arbitrary trace
  - Let  $(tr \downarrow choc)$  denote  $\#(tr \uparrow \{ choc \})$ 
    - Number  $choc$  events in trace  $tr$
- **Example**
  - **VM owner:** # of chocolate must never exceed # of coins inserted
    - $NOLOSS = (tr \downarrow choc) \leq (tr \downarrow coin)$
  - **VM customers:** VM won't take more coins until it dispenses paid chocolate
    - $FAIR = (tr \downarrow coin) \leq (tr \downarrow choc) + 1$
  - Hence, **VM manufacturer** must meet spec. from both VM owner and customers
    - $VMSPEC = NOLOSS \wedge FAIR$   
 $= 0 \leq (tr \downarrow coin) - (tr \downarrow choc) \leq 1$

CIS 505, Spring 2007

CSP

18

## Satisfaction

- **$P$  sat  $S$** 
  - If a product  $P$  **meets a specification**  $S$ , then  $P$  satisfies  $S$
  - Formally,  $P$  sat  $S$  iff  $\forall tr \bullet tr \in traces(P) \Rightarrow S$
- **Example:**  $VMC$  sat  $VMSPEC$ 
  - Recall:
    - $VMC = (coin \rightarrow (choc \rightarrow VMC))$
    - $VMSPEC = NOLOSS \wedge FAIR$   
 $= 0 \leq (tr \downarrow coin) - (tr \downarrow choc) \leq 1$

CIS 505, Spring 2007

CSP

19

## Concurrency

- **2 or more processes operating together**
  - **Syntax:**  $P \parallel Q$
  - Example: Both customers and vending machines can be viewed as processes interacting with one another
- **Interaction**
  - Interact via shared events between processes
- **Concurrency**
  - Specifies how shared and private events in processes are joined

CIS 505, Spring 2007

CSP

20

## Interaction Example

- A **greedy** customer tries to get chocolate or toffee without paying
- If doesn't work, he reluctantly pays for chocolate
  - $GRCUST = (toffee \rightarrow GRCUST$   
 $\quad | choc \rightarrow GRCUST$   
 $\quad | coin \rightarrow choc \rightarrow GRCUST)$
- When using VMCT machine,
  - $VMCT = \mu X \cdot coin \rightarrow (choc \rightarrow X | toffee \rightarrow X)$
- he can't get goods without paying, hence, he only gets chocolate
  - $(GRCUST || VMCT) = \mu X \cdot (coin \rightarrow choc \rightarrow X)$

CIS 505, Spring 2007

CSP

21

## Interaction Laws and Traces

- **Laws:**
  - $P || Q = Q || P$
  - $P || (Q || R) = (P || Q) || R$
  - $P || STOP_{\alpha P} = STOP_{\alpha P}$
  - $P || RUN_{\alpha P} = P$
  - $(c \rightarrow P) || (c \rightarrow Q) = (c \rightarrow (P || Q))$
  - $(c \rightarrow P) || (d \rightarrow Q) = STOP$  if  $c \neq d$
- **Traces:**
  - $traces(P || Q) = traces(P) \cap traces(Q)$
  - $(P || Q) / s = (P / s) || (Q / s)$

CIS 505, Spring 2007

CSP

22

## Concurrency

- In general, it's possible that
  - $\alpha P \neq \alpha Q$
  - for  $x \in (\alpha P - \alpha Q)$ ,
    - $P$  may engage alone with no concern to  $Q$
    - and vice versa
- Hence,
  - $\alpha(P || Q) = \alpha P \cup \alpha Q$

CIS 505, Spring 2007

CSP

23

## Concurrency Example

- **Noisy** vending machine
  - $\alpha NOISYVM = \{coin, choc, clink, clunk, toffee\}$
  - $NOISYVM = (coin \rightarrow clink \rightarrow choc \rightarrow clunk \rightarrow NOISYVM)$
- Customer who **curse**s when gets chocolate instead of toffee
  - $\alpha CUST = \{coin, choc, curse, toffee\}$
  - $CUST = (coin \rightarrow (toffee \rightarrow CUST$   
 $\quad | curse \rightarrow choc \rightarrow CUST)$
- **Cursing** customer using **noisy** machine
  - $(NOISY || CUST) =$   
 $\mu X \cdot (coin \rightarrow (clink \rightarrow curse \rightarrow choc \rightarrow clunk \rightarrow X$   
 $\quad | curse \rightarrow clink \rightarrow choc \rightarrow clunk \rightarrow X))$

CIS 505, Spring 2007

CSP

24

## Concurrency Law

- $\parallel$  is symmetric and associative
- $P \parallel STOP_{\alpha P} = STOP_{\alpha P}$
- $P \parallel RUN_{\alpha P} = P$
- Let
  - $a \in (\alpha P - \alpha Q)$
  - $b \in (\alpha Q - \alpha P)$
  - $\{c, d\} \subseteq (\alpha P \cap \alpha Q)$
- Then
  - $(c \rightarrow P) \parallel (c \rightarrow Q) = c \rightarrow (P \parallel Q)$
  - $(c \rightarrow P) \parallel (d \rightarrow Q) = STOP$  if  $c \neq d$
  - $(a \rightarrow P) \parallel (c \rightarrow Q) = a \rightarrow (P \parallel (c \rightarrow Q))$
  - $(c \rightarrow P) \parallel (b \rightarrow Q) = b \rightarrow ((c \rightarrow P) \parallel Q)$
  - $(a \rightarrow P) \parallel (b \rightarrow Q) = (a \rightarrow (P \parallel (b \rightarrow Q))) \mid (b \rightarrow ((a \rightarrow P) \parallel Q))$

CIS 505, Spring 2007

CSP

25

## Concurrency Traces

- $traces(P \parallel Q)$  is **all possible** traces of process  $(P \parallel Q)$

$$traces(P \parallel Q) = \{ t \mid (t \upharpoonright \alpha P) \in traces(P) \wedge (t \upharpoonright \alpha Q) \in traces(Q) \wedge t \in (\alpha P \cup \alpha Q)^* \}$$

traces of all events where P participates  
traces of all events where Q participates  
every event in t must be in either  $\alpha P$  or  $\alpha Q$

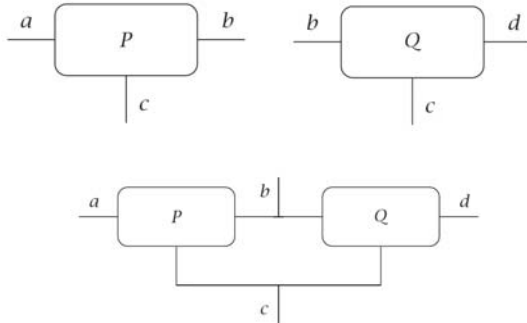
- $t1 = \langle coin, clink, curse \rangle \in traces(NOISYVM \parallel CUST)$ 
  - $t1 \upharpoonright \alpha NOISYVM = \langle coin, clink \rangle \in traces(NOISYVM)$
  - $t1 \upharpoonright \alpha CUST = \langle coin, curse \rangle \in traces(CUST)$
- Similar for  $t2 = \langle coin, curse, clink \rangle$

CIS 505, Spring 2007

CSP

26

## Concurrency Pictures

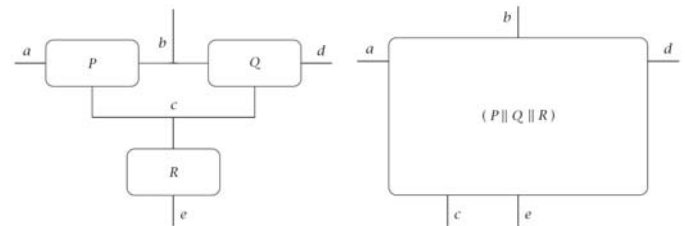


CIS 505, Spring 2007

CSP

27

## Concurrency Pictures



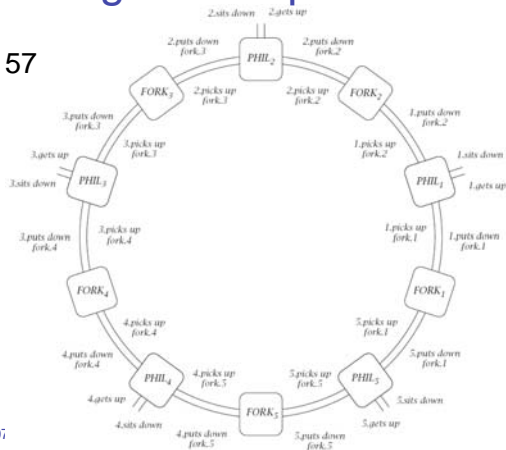
CIS 505, Spring 2007

CSP

28

## Dining Philosophers

- Book p. 57



CIS 505, Spring 2007

19

## Change of Symbol

- Define groups of processes with **similar behaviors**
  - One-on-one function (injection) maps alphabet of P onto a set of symbols A
  - $f: \alpha P \rightarrow A$
- Example:
  - Recall: **Chocolate / toffee** machine
    - $VMCT = coin \rightarrow (choc \rightarrow VMCT \mid toffee \rightarrow VMCT)$
  - Want: **Gum / pretzel** machine
    - $f(coin) = coin$
    - $f(choc) = gum$
    - $f(toffee) = pretzel$
  - Now,  $VMGP = f(VMCT)$

CIS 505, Spring 2007

CSP

30

## Process Labeling

- Label **identical but independent** processes
- Example:
  - Two machine standing side by side
    - $(left : VMC) \parallel (right : VMC)$
  - Possible trace
    - $\langle left.coin, right.coin, right.choc \rangle$

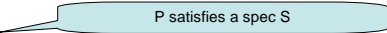
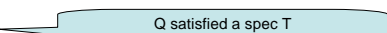
CIS 505, Spring 2007

CSP

31

## Specifications and Satisfaction

- Satisfaction for concurrent processes

- **If**  $P \text{ sat } S(tr)$  
- and**  $Q \text{ sat } T(tr)$  
- then**  $(P \parallel Q) \text{ sat } S(tr \uparrow \alpha P) \wedge T(tr \uparrow \alpha Q)$

$P \parallel Q$  satisfies if in trace  $tr$

- events of P satisfy S
- events of Q satisfy T

CIS 505, Spring 2007

CSP

32

## Spec, Sat Example

- Let  $\alpha P = \{a, c\}$  and  $\alpha Q = \{b, c\}$ 
  - $P = (a \rightarrow c \rightarrow P)$
  - $Q = (c \rightarrow b \rightarrow Q)$
- **Want to know** if  $(P \parallel Q) \text{ sat } 0 \leq (\text{tr} \downarrow a) - (\text{tr} \downarrow b) \leq 2$ 

num of a and b differ at most 2
- **Obviously,**
  - $P \text{ sat } 0 \leq (\text{tr} \downarrow a) - (\text{tr} \downarrow c) \leq 1$ 

In P, num of a and c differ at most 1
  - $Q \text{ sat } 0 \leq (\text{tr} \downarrow c) - (\text{tr} \downarrow b) \leq 1$ 

In Q, num of c and b differ at most 1
- **Hence,**
  - $(P \parallel Q) \text{ sat } (0 \leq (\text{tr} \uparrow \alpha P) \downarrow a - (\text{tr} \uparrow \alpha P) \downarrow c \leq 1 \wedge 0 \leq (\text{tr} \uparrow \alpha Q) \downarrow c - (\text{tr} \uparrow \alpha Q) \downarrow b \leq 1)$
  - Since  $(\text{tr} \uparrow A) \downarrow a = \text{tr} \downarrow a$  when  $a \in A$ 

num of a when trace has only events in A and  $a \in A$
  - $(P \parallel Q) \text{ sat } (0 \leq (\text{tr} \downarrow a) - (\text{tr} \downarrow c) \leq 1 \wedge 0 \leq (\text{tr} \downarrow c) - (\text{tr} \downarrow b) \leq 1)$
  - $(P \parallel Q) \text{ sat } (0 \leq (\text{tr} \downarrow a) - (\text{tr} \downarrow b) \leq 2)$

## Theory of Deterministic Processes

- CSP Book P. 72 – 79
- It shows that
  - CSP laws are in fact true
  - a recursively defined process is indeed a solution of the corresponding recursive equation (fixed point theory)
  - there exists a unique solution