The Interplay of Machine Learning and Mechanism Design

David C. Parkes
Harvard University

learn a hypothesis given a distribution on inputs (and outputs)

$h: X \to Y$

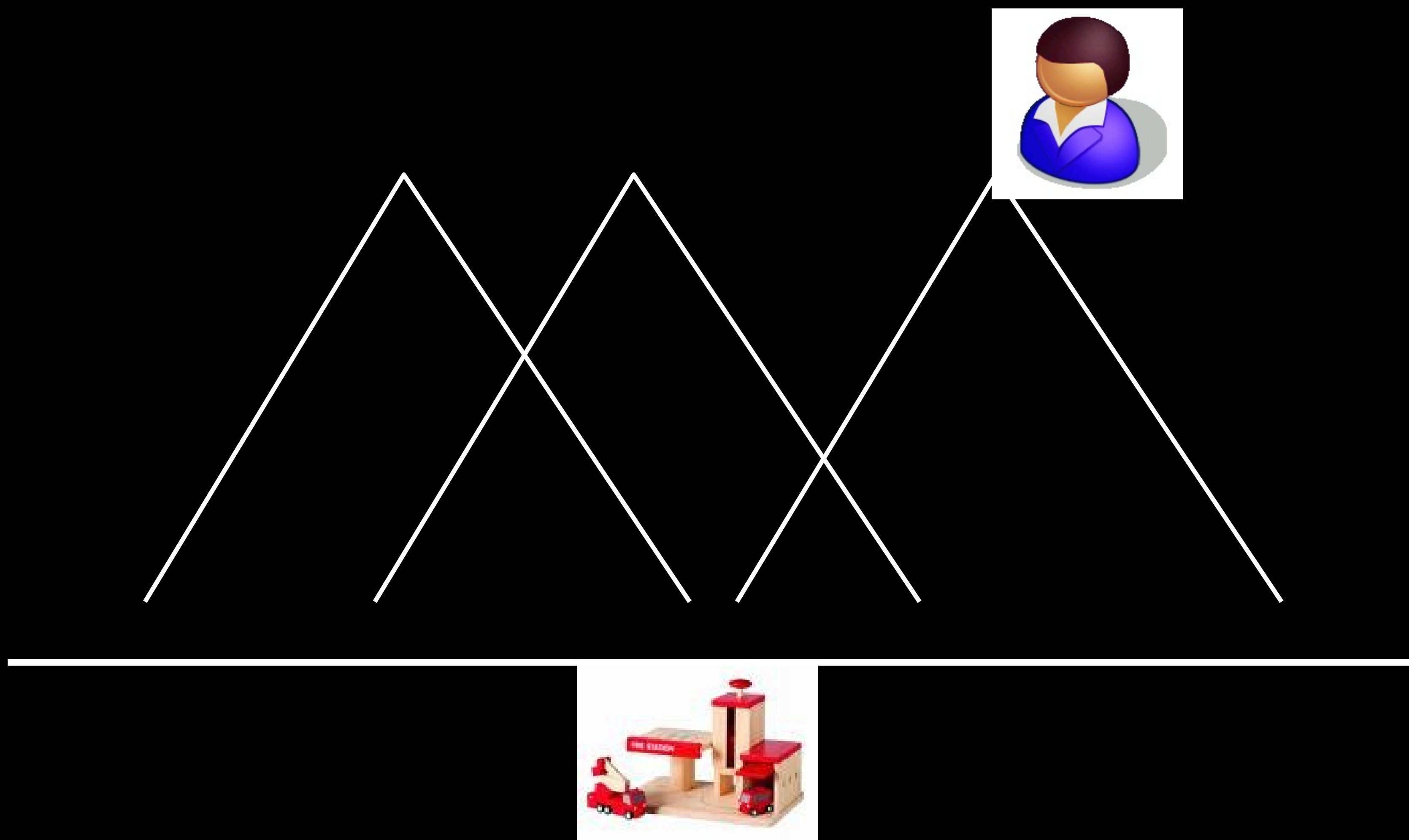
design a decision rule to use on reports of private inputs

$g: X^n \to Y$

incentive compatibility

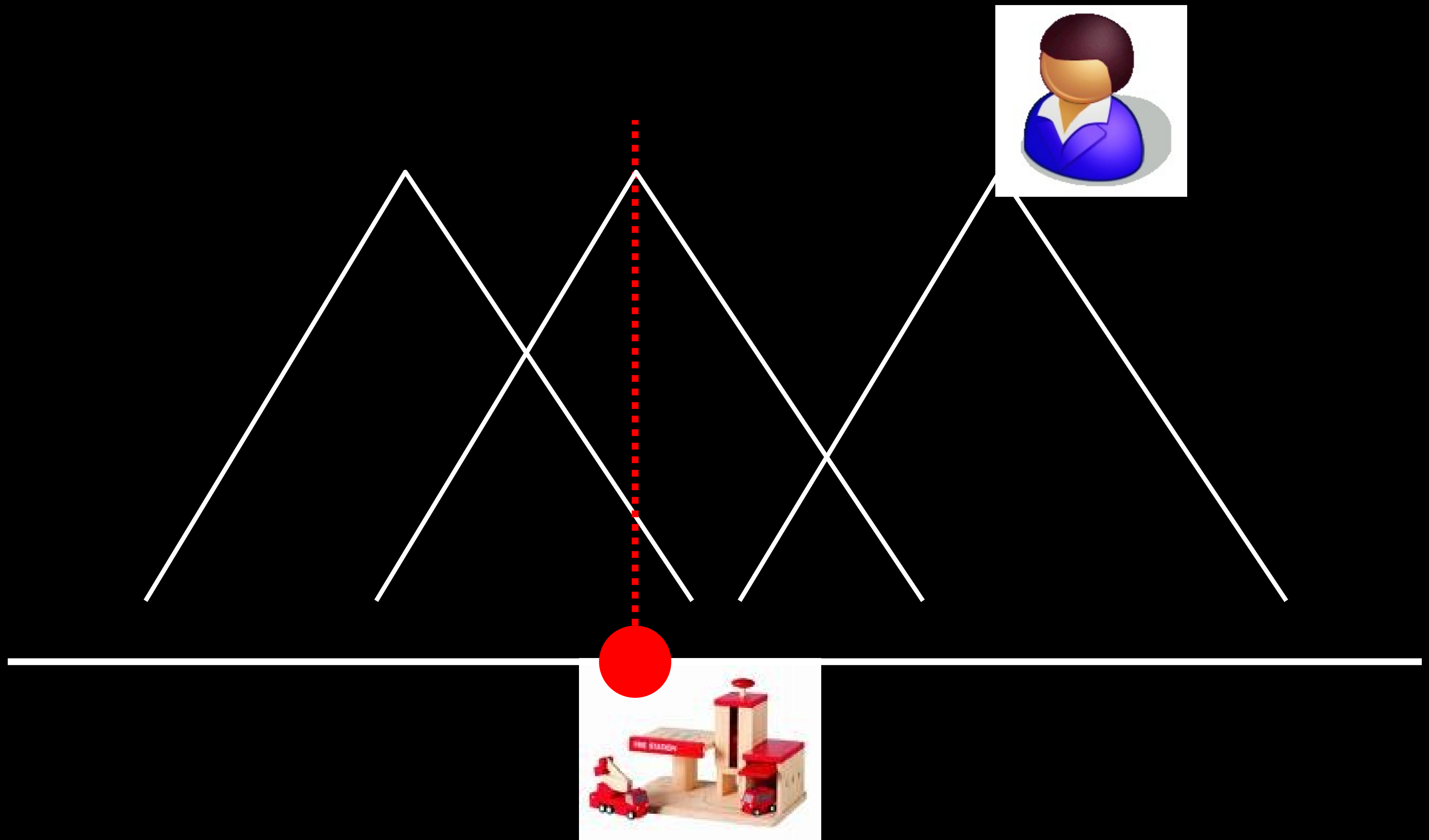
Example: Single-peaked preferences

(Moulin'80)



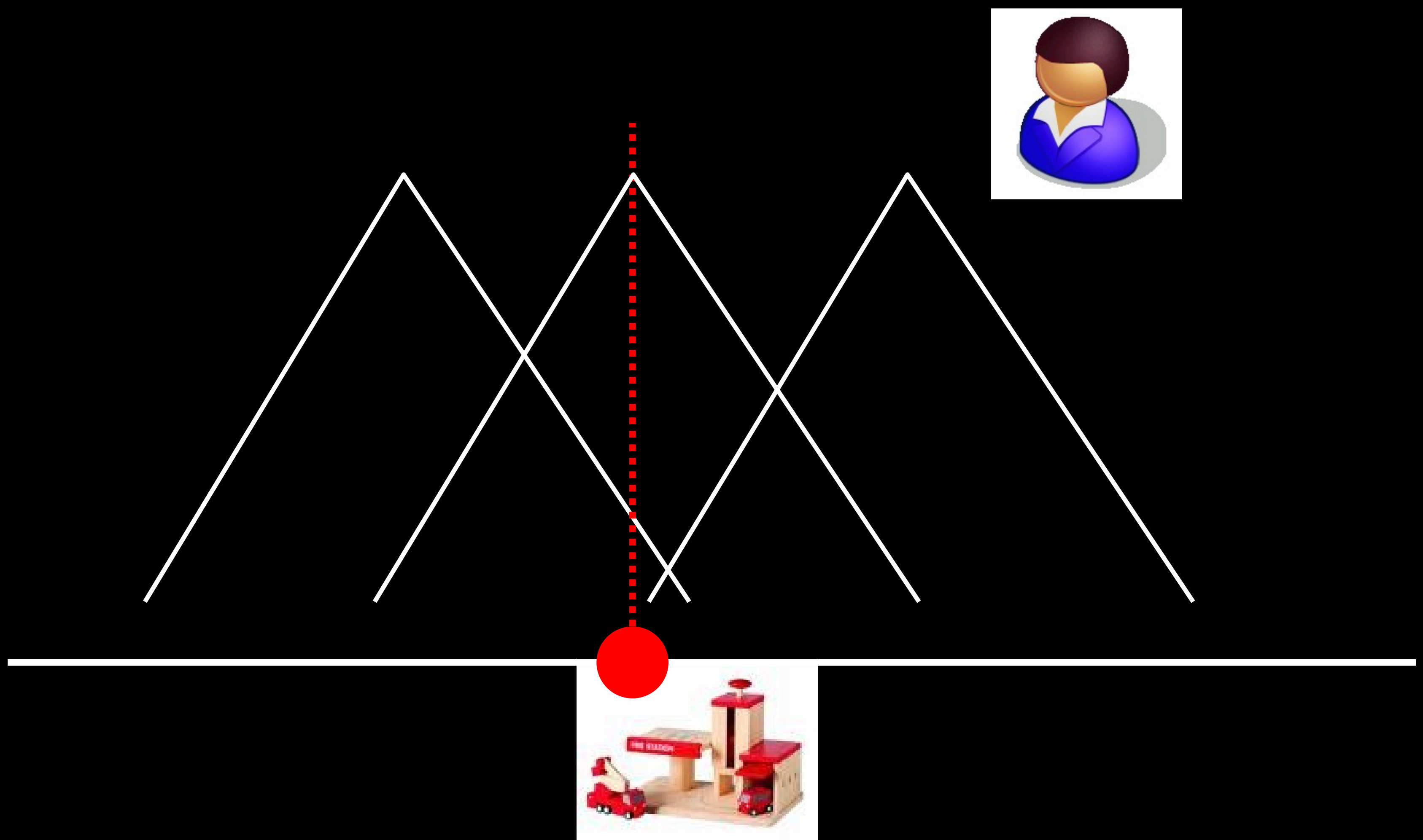
Example: Single-peaked preferences

(Moulin'80)

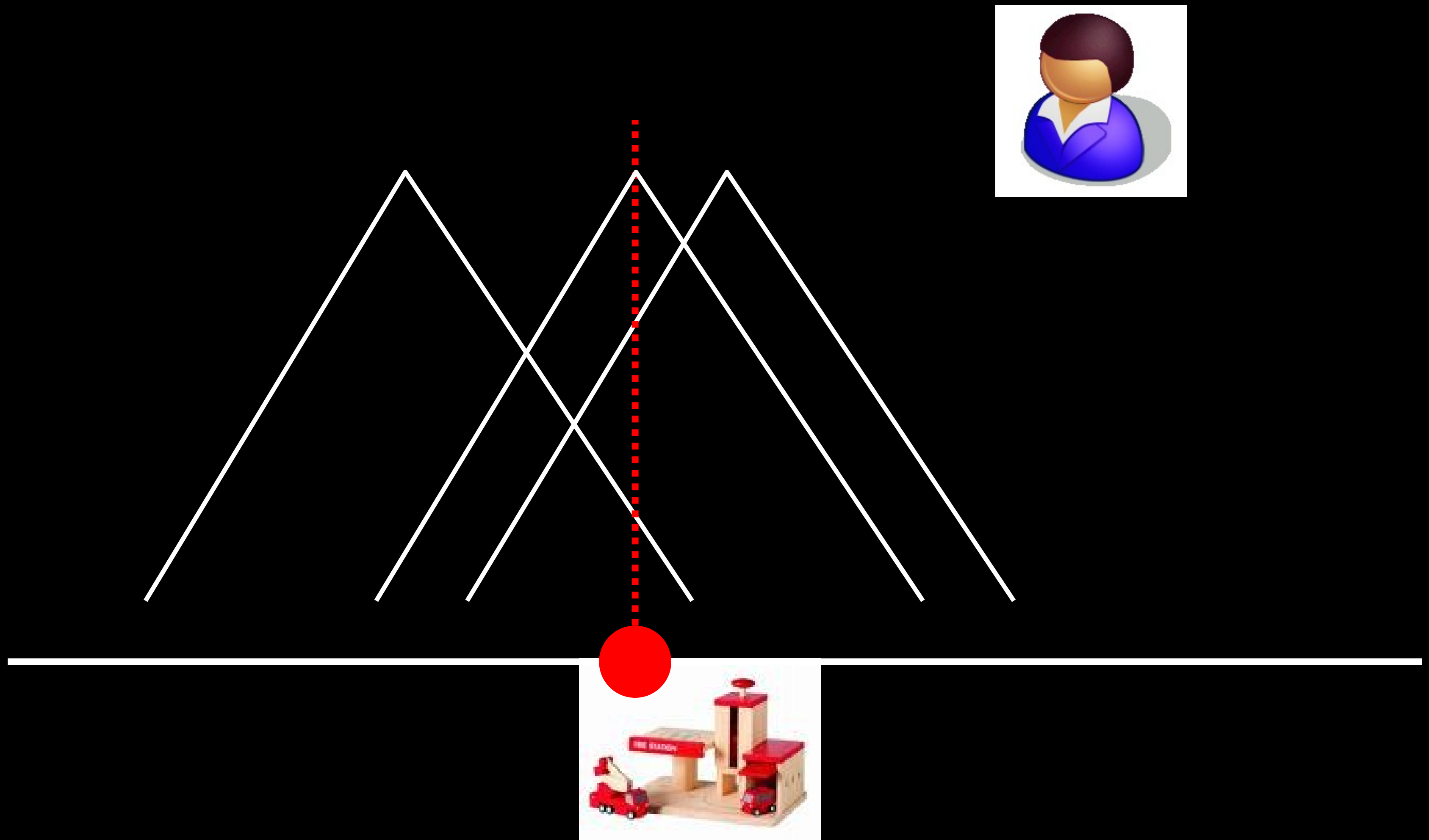


Example: Single-peaked preferences

(Moulin'80)

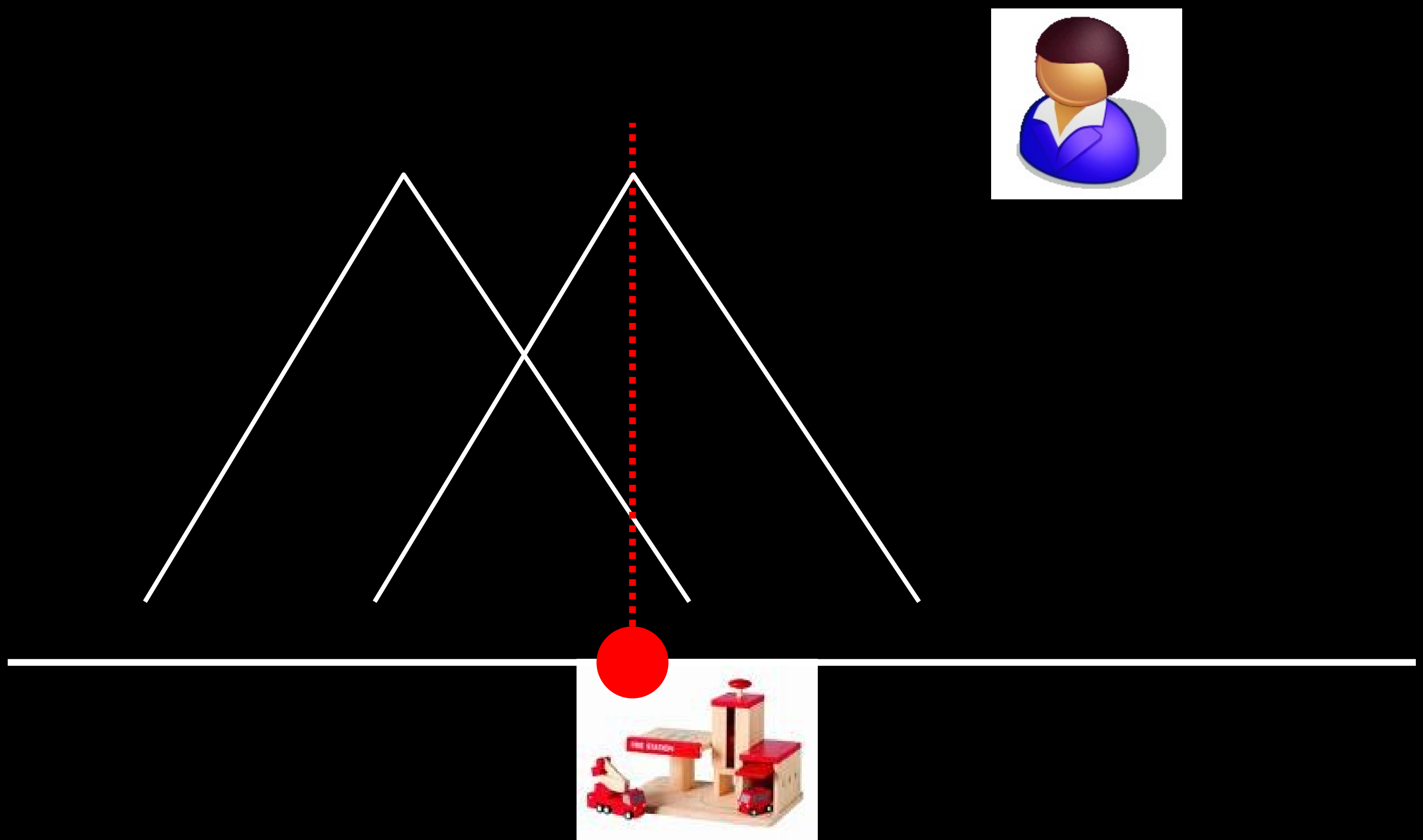


Example: Single-peaked preferences
(Moulin'80)

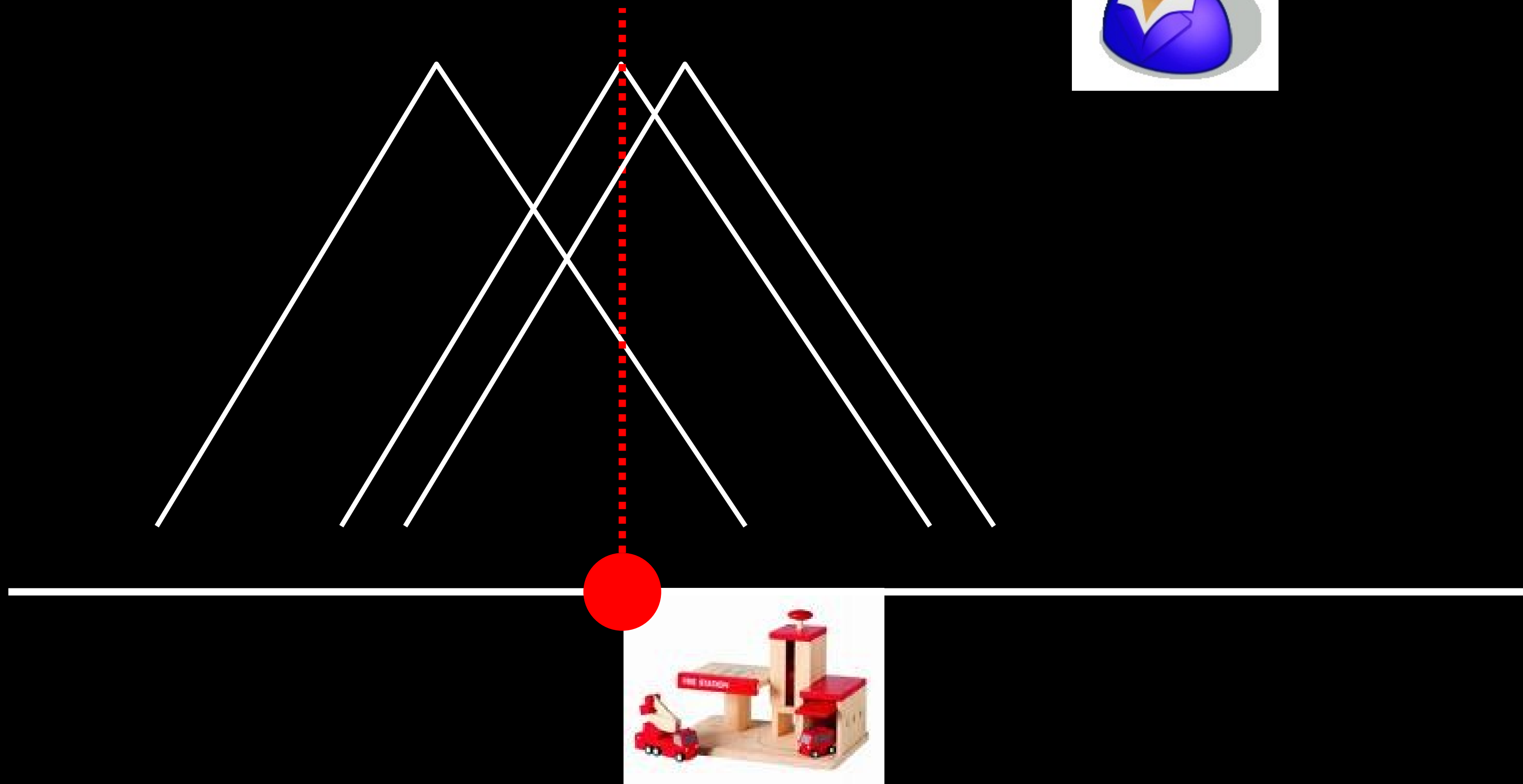


Example: Single-peaked preferences

(Moulin'80)

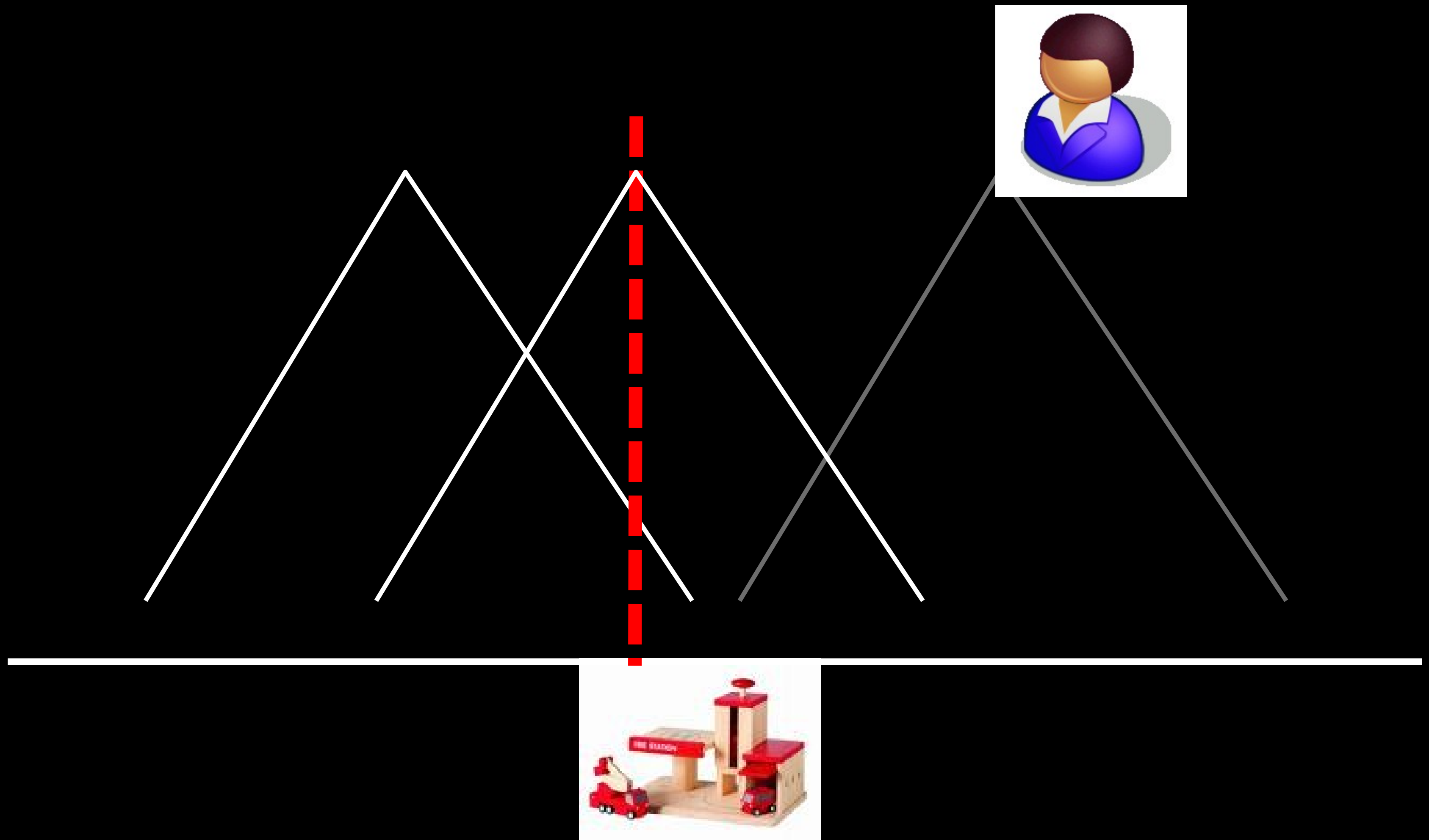


Example: Single-peaked preferences
(Moulin'80)



Example: Single-peaked preferences

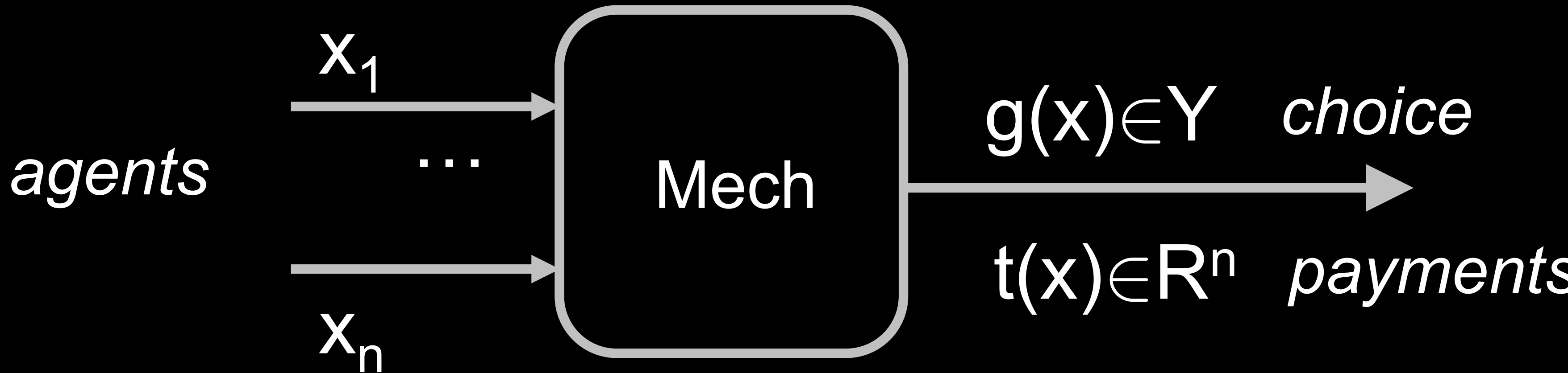
(Moulin'80)



Single-item auction

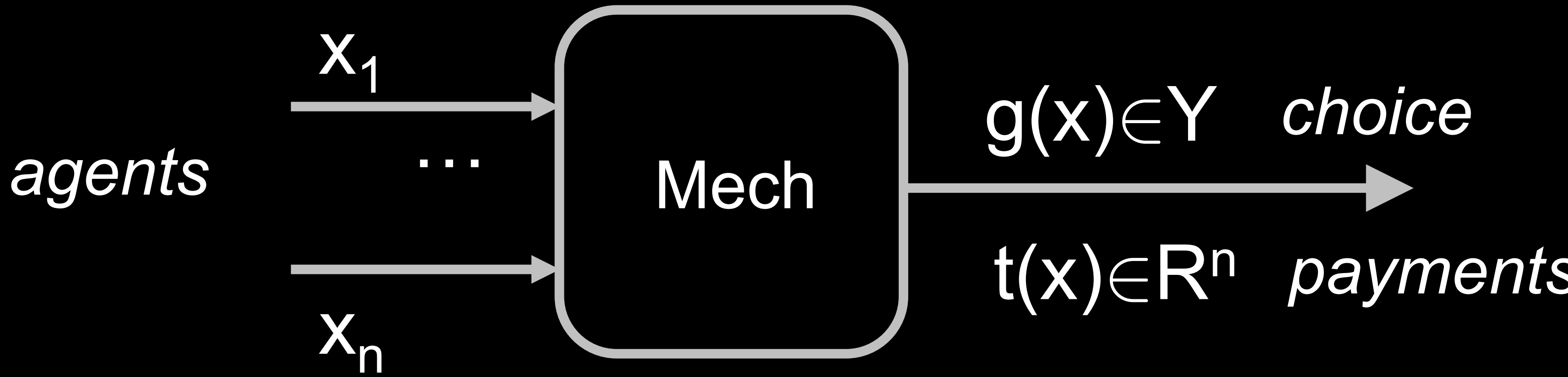
(Vickrey'61)

A mechanism



value $x_i[y] \in R$

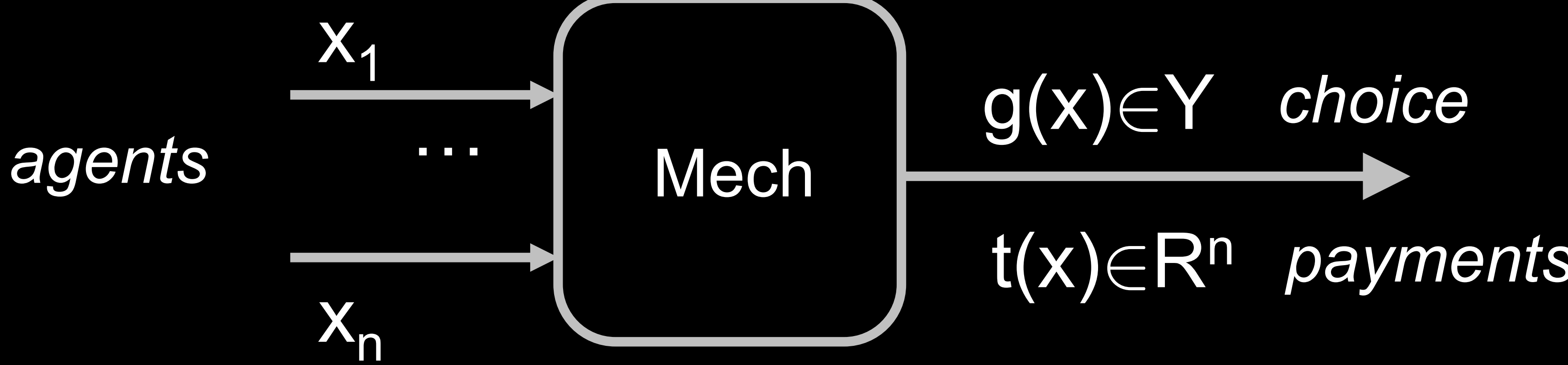
A mechanism



value $x_i[y] \in R$. Incentive compatibility:

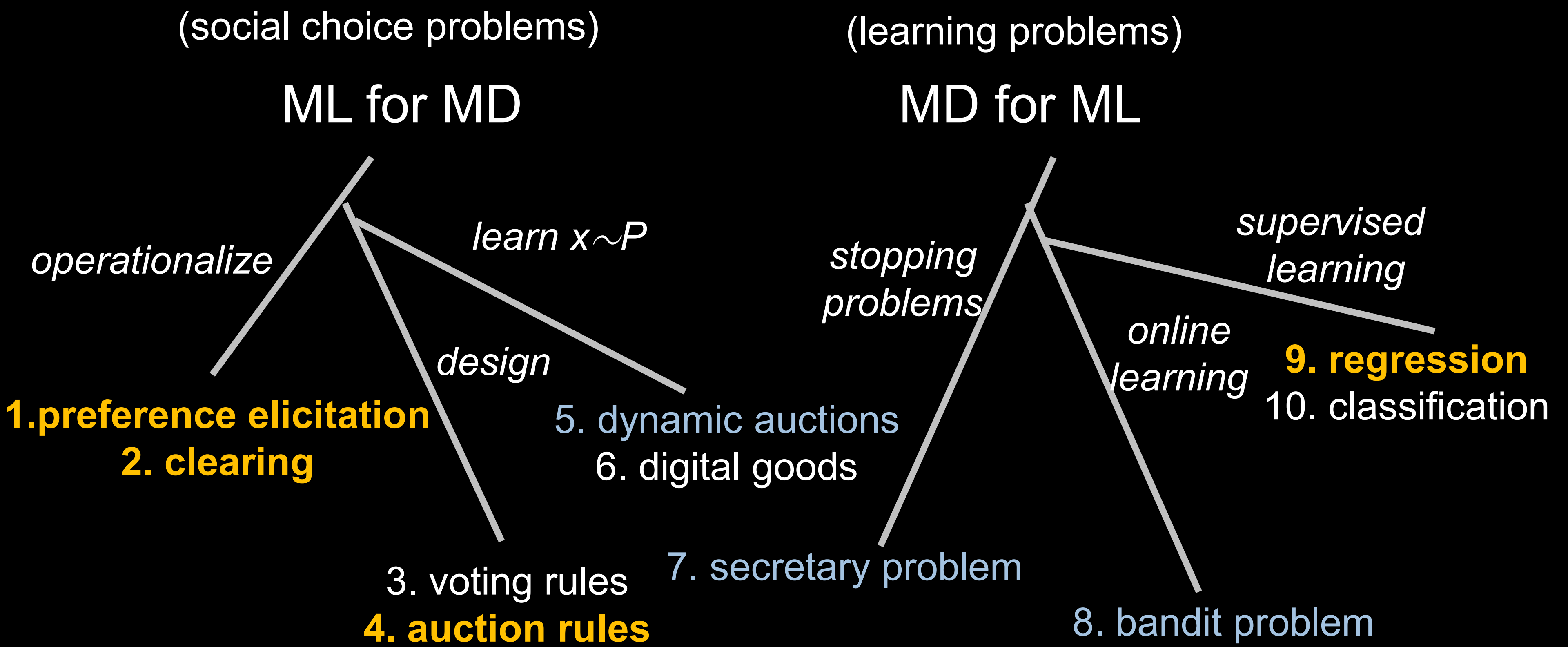
$x_i[g(x_i x_{-i})] - t_i(x_i,x_{-i}) \geq$

A mechanism



value $x_i[y] \in R$. Incentive compatibility:

$$x_i[g(x_i x_{-i})] - t_i(x_i,x_{-i}) \geq x_i[g(x_i',x_{-i})] - t_i(x_i',x_{-i}) \quad \forall i, \forall x_i, \forall x_{-i}, \forall x_i'$$


(social choice problems)
ML for MD
operationalize
learn x∼P
design
1.preference elicitation
2. clearing
5. dynamic auctions
6. digital goods
3. voting rules
4. auction rules
(learning problems)
MD for ML
stopping problems
supervised learning
online learning
9. regression
10. classification
7. secretary problem
8. bandit problem

(social choice problems)

ML for MD

(learning problems)

MD for ML

operationalize

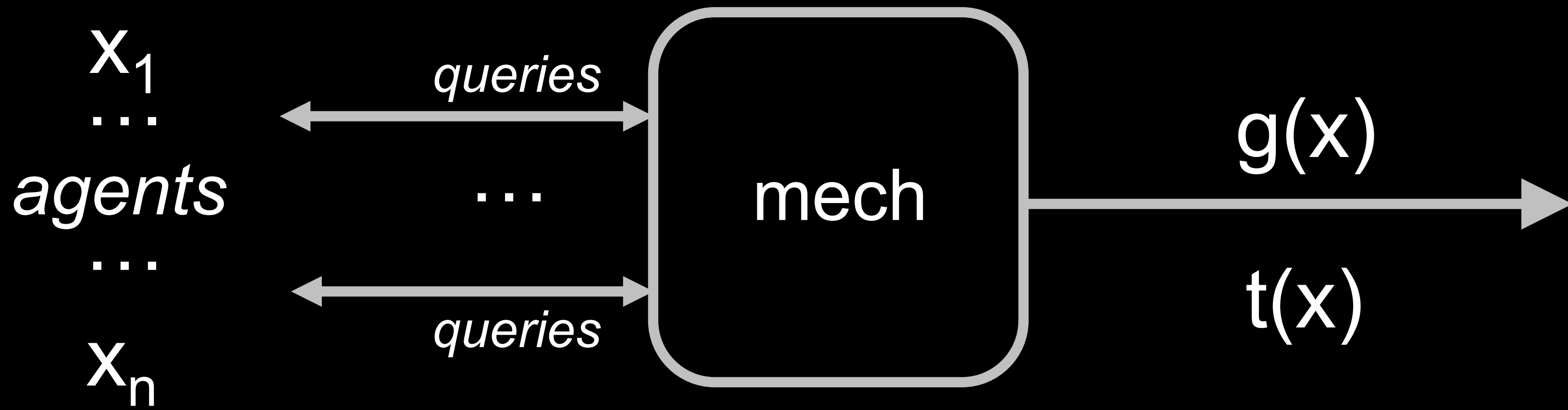
1.preference elicitation

1. Pref Elicit for Comb. Auctions

- m goods, n agents
- x_i: $\{0,1\}^m \to R$
  - complements, substitutes

1. Pref Elicit for Comb. Auctions

- m goods, n agents
- $x_i: \{0,1\}^m \rightarrow R$
  - complements, substitutes



Bidding Language

- $\mathcal{B}$: { (A, 10), (B, 12), (BC, 20)}
  - set of (bundle, value) pairs

Bidding Language

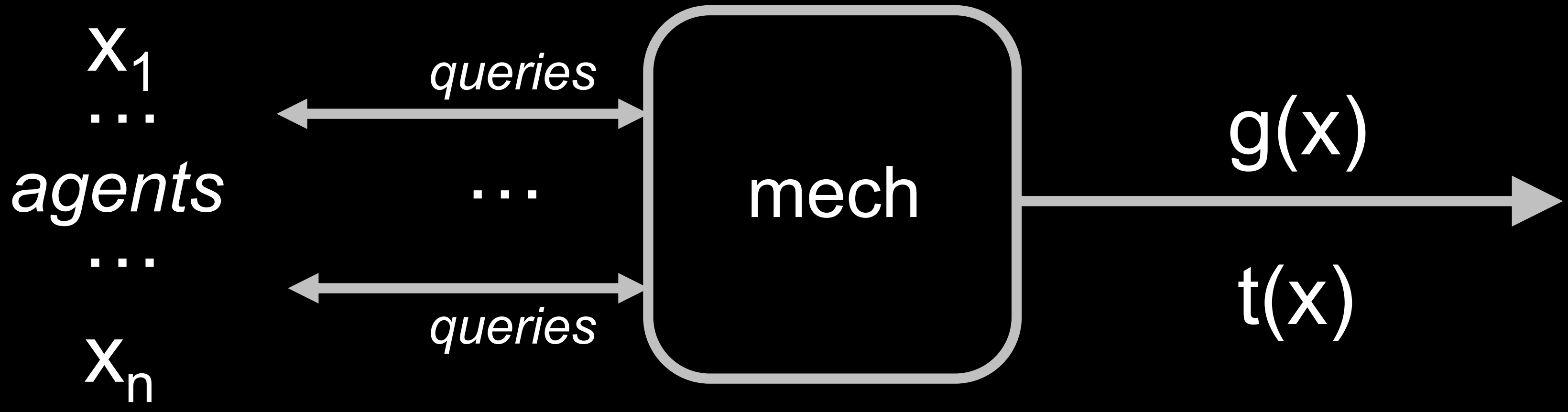
- $\mathcal{B}$: { (A, 10), (B, 12), (BC, 20)}
  - set of (bundle, value) pairs
- L_{XOR}: $x_i(AB) = 12$; $x_i(ABC) = 20$

Bidding Language

- $\mathcal{B}$: { (A, 10), (B, 12), (BC, 20)}
  - ❖ set of (bundle, value) pairs
- L_{XOR}: $x_i(AB) = 12$; $x_i(ABC) = 20$
- L_{OR}: $x_i(AB) = 22$; $x_i(ABC) = 30$

… other languages

1. Pref Elicit for Comb. Auctions



size$_L$(x_i): minimal |$\mathcal{B}$| to represent x_i in L

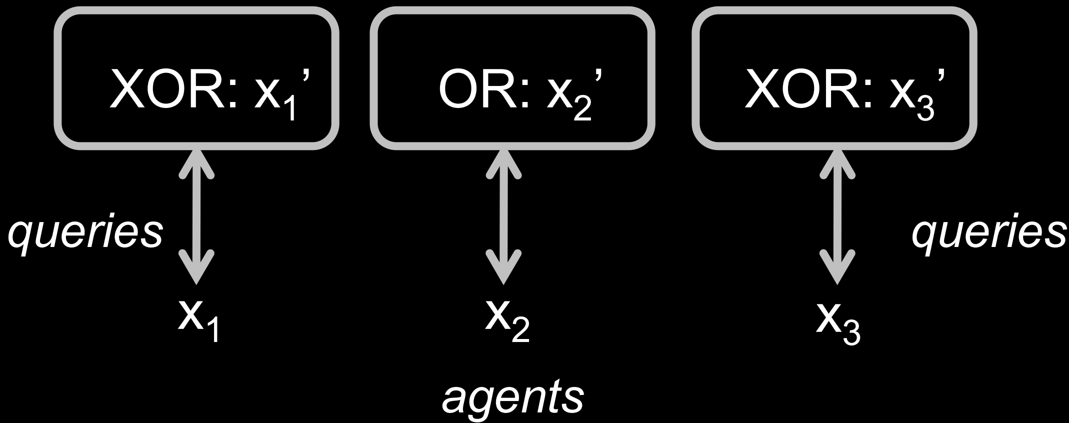
Goal 1: Exact query learning with *value* and (linear) *demand* queries
#queries poly in size, m and n

Goal 2: Determine outcome with *value* and *demand* queries
#queries poly in size, m, and n

ML4MD:1

1. Pref Elicit for Comb. Auctions

(Lahaie & P. EC'04)

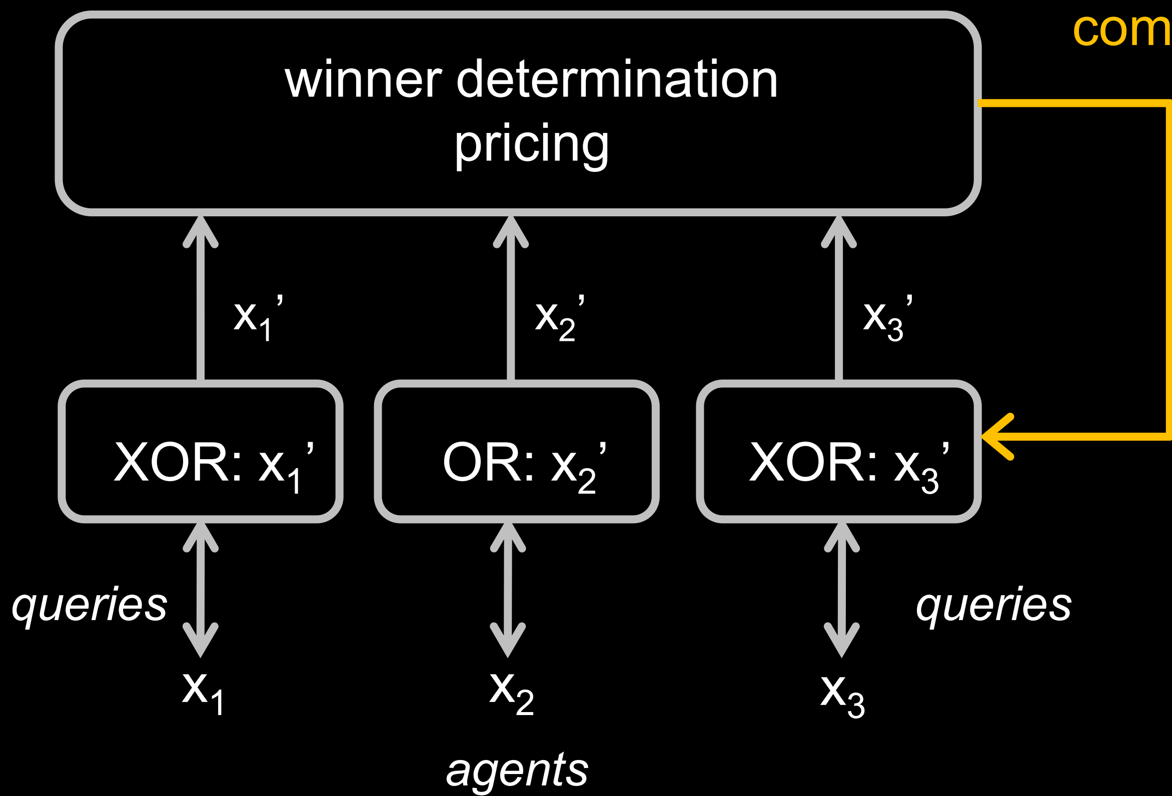


equivalence:
"is $h(x) = y$"?
Yes, or "no, $h(x)=y'$."
m'ship: "what is $f(x)$"?

demand:
"does y maximize
utility given p"?
Yes, or "no, y' does."
value: "what is $x_i[y]$"?

1. Pref Elicit for Comb. Auctions

(Lahaie & P. EC'04)



competitive equilibrium
(y',p') given x'

equivalence:
"is h(x) = y"?
Yes, or "no, h(x)=y'."
m'ship: "what is f(x)"?

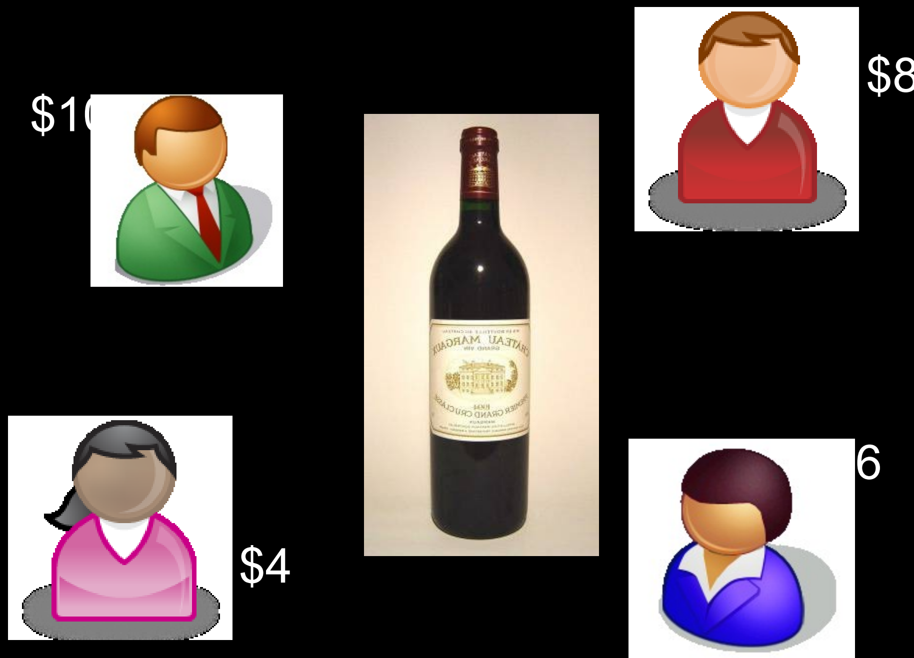
demand:
"does y maximize
utility given p"?
Yes, or "no, y' does."
value: "what is x_i[y]"?

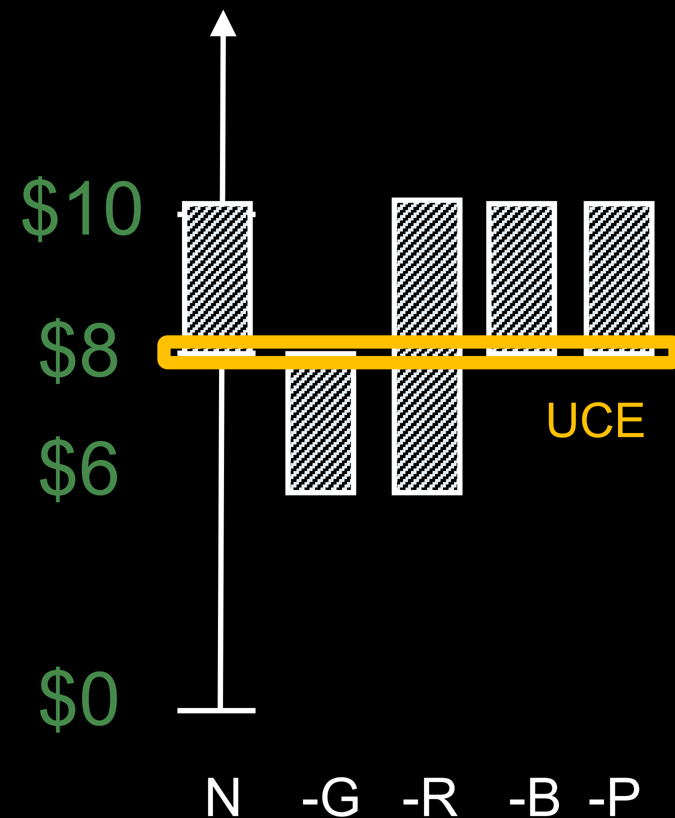
polynomial-query learning ⇒ *polynomial-query elicitation*

modular framework

What about self-interest?

(Constantin, Lahaie, P. AAAI‘05)





Thm. Communication protocol for UCE

⇔ Communication protocol for VCG

Idea: simulate queries until get to UCE

ML4MD:1

(social choice problems)

ML for MD

(learning problems)

MD for ML

operationalize

1. preference elicitation
2. clearing

2. Kernel Methods for Clearing

(Lahaie'09, Lahaie'10)

- Standard: $(x_1,\ldots,x_n) \rightarrow$ determine allocation and payments by solving n+1 problems.

2. Kernel Methods for Clearing

(Lahaie'09, Lahaie'10)

- Standard: $(x_1,\ldots,x_n) \rightarrow$ determine allocation and payments by solving n+1 problems.

Kernel method:

- Allocation and payments in one step

2. Kernel Methods for Clearing

(Lahaie'09, Lahaie'10)

- Standard: $(x_1,\ldots,x_n) \rightarrow$ determine allocation and payments by solving n+1 problems.

Kernel method:

- Allocation and payments in one step
- Non-linear prices as linear prices
  - $p(y) = w^T\phi(y)$, $y \in \{0,1\}^m = Y$, $\phi: Y \rightarrow R^M$
  - kernels $\sim$ different price spaces

2. Kernel Methods for Clearing

(Lahaie'09, Lahaie'10)

- Standard: $(x_1,\ldots,x_n) \rightarrow$ determine allocation and payments by solving n+1 problems.

Kernel method:

- Allocation and payments in one step
- Non-linear prices as linear prices
  - $p(y) = w^T\phi(y)$, $y\in\{0,1\}^m = Y$, $\phi:Y\rightarrow R^M$
  - kernels $\sim$ different price spaces
- Connect stability and UCE, thus incentives

$$\max_{\alpha \geq 0,\ \beta \geq 0} \sum_i \alpha_i x_i$$

$$\text{s.t.} \quad \alpha_i \leq 1$$

winner determination

c.f. SVM dual

single-minded

(x_i, y_i)

$$\max_{\alpha \geq 0,\ \beta \geq 0} \ \sum_i \alpha_i x_i - \frac{1}{2\lambda} \left\| \sum_i \Big(\alpha_i - \sum_{j:\, y_i \in A_j} \beta_k\Big)\phi(y_i) \right\|^2$$

$$\text{s.t.} \quad \alpha_i \leq 1; \ \sum_j \beta_j \leq 1$$

winner determination
c.f. SVM dual

$$\min_{\pi \geq 0,\ \pi_o \geq 0,\ w} \ \sum_i \pi_i + \pi_o + \frac{\lambda}{2} w^T w$$

$$\text{s.t.} \quad \pi_i \geq x_i - w^T\phi(y_i), \quad \forall i$$

$$\pi_o \geq \sum_{i:\, y_i \in Y_j} w^T\phi(y_i), \quad \forall j$$

pricing
c.f. SVM primal

single-minded
(x_i, y_i)

Regularization: *more stable*, and closer to UCE prices!

ML4MD:2

Stability: Incentive analysis (Lahaie'10)

- If w is optimal dual solution and w_{-i} is optimal without i, then $||w - w_{-i}|| \leq \frac{\kappa}{\lambda}$
- Obtain ϵ-SP for $\epsilon = \frac{2(n-1)\,\kappa^2}{\lambda}$
- More complex ($\kappa\Downarrow$), more regularization ($\lambda\uparrow$), closer to UCE and IC (tradeoff w/ feasibility)

(social choice problems)

ML for MD

operationalize

design

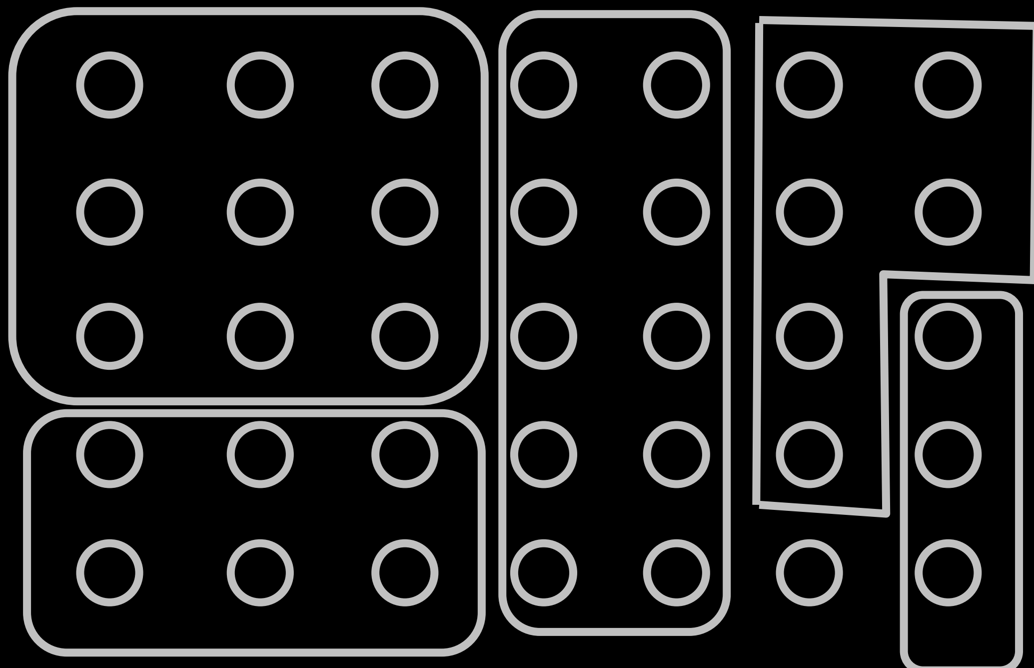
1. preference elicitation
2. clearing

3. voting rules
4. auction rules

(learning problems)

MD for ML

4. SVMs for Defining Payment Rules



Problem statement

- Given an allocation algorithm g: X$\rightarrow$ Y, find a payment rule t: X$\rightarrow$ R^n that is “maximally incentive compatible”

Problem statement

- Given an allocation algorithm g: $X \rightarrow Y$, find a payment rule t: $X \rightarrow R^n$ that is "maximally incentive compatible"

- Solution idea: Generate $(x,y) \sim P(X,Y)$. Train a classifier. Use to price.

Example: Single-item allocation

- $X = R^n$; $g: X \to \{\pm 1\}$
- Inputs: ((10,8,7), 1), ((5,8,7), -1), ((9,2,5), +1)
- Learn $f: X \to R$; $h(x) = sgn(f(x))$

Example: Single-item allocation

- $X = R^n$; $g: X \to \{\pm 1\}$
- Inputs: ((10,8,7), 1), ((5,8,7), -1), ((9,2,5), +1)
- Learn $f: X \to R$; $h(x) = \text{sgn}(f(x))$
- Exact classifier: $f(x) = x_1 - \max(x_{-1})$

Example: Single-item allocation

- $X = R^n$; $g: X \rightarrow \{\pm 1\}$
- Inputs: ((10,8,7), 1), ((5,8,7), -1), ((9,2,5), +1)
- Learn $f: X \rightarrow R$; $h(x) = \mathrm{sgn}(f(x))$
- Exact classifier: $f(x) = x_1 - \max(x_{-1})$
- Require: $f(x) = x_1 + w^T \phi(x_{-1}) = x_1 - t_1(x_{-1})$
  - discriminant $\equiv$ payoff

Example: Single-item allocation

- $X = R^n$; $g: X \rightarrow \{\pm 1\}$
- Inputs: ((10,8,7), 1), ((5,8,7), -1), ((9,2,5), +1)
- Learn $f: X \rightarrow R$; $h(x) = \mathrm{sgn}(f(x))$
- Exact classifier: $f(x) = x_1 - \max(x_{-1})$
- Require: $f(x)= x_1 + w^T \phi(x_{-1}) = x_1 - t_1(x_{-1})$
  - discriminant $\equiv$ payoff

- SP: If $x_1 - t_1(x_{-1}) \geq 0$, then $g(x) =1$
  $x_1 - t_1(x_{-1}) < 0$, then $g(x) = -1$

General problem

(Duetting, Fischer, Jirapinyo, Lai, Lubin and P.'10)

- $x \in R^{2^m \times n}$; $y \in \{0,1\}^m$
- Learn $h: X \to Y$; $h(x) = \arg\max_y f(x,y)$

General problem

(Duetting, Fischer, Jirapinyo, Lai, Lubin and P.'10)

- $x \in R^{2^m \times n}$; $y \in \{0,1\}^m$
- Learn h: $X \rightarrow Y$; $h(x) = \arg\max_y f(x,y)$
- Stipulate $f(x,y) = x_1[y] + w^T \psi(x_{-1}, y)$
- Payment: $t_1(x,y) = - w^T \psi(x_{-1}, y)$
  - discriminant $\equiv$ payoff

General problem

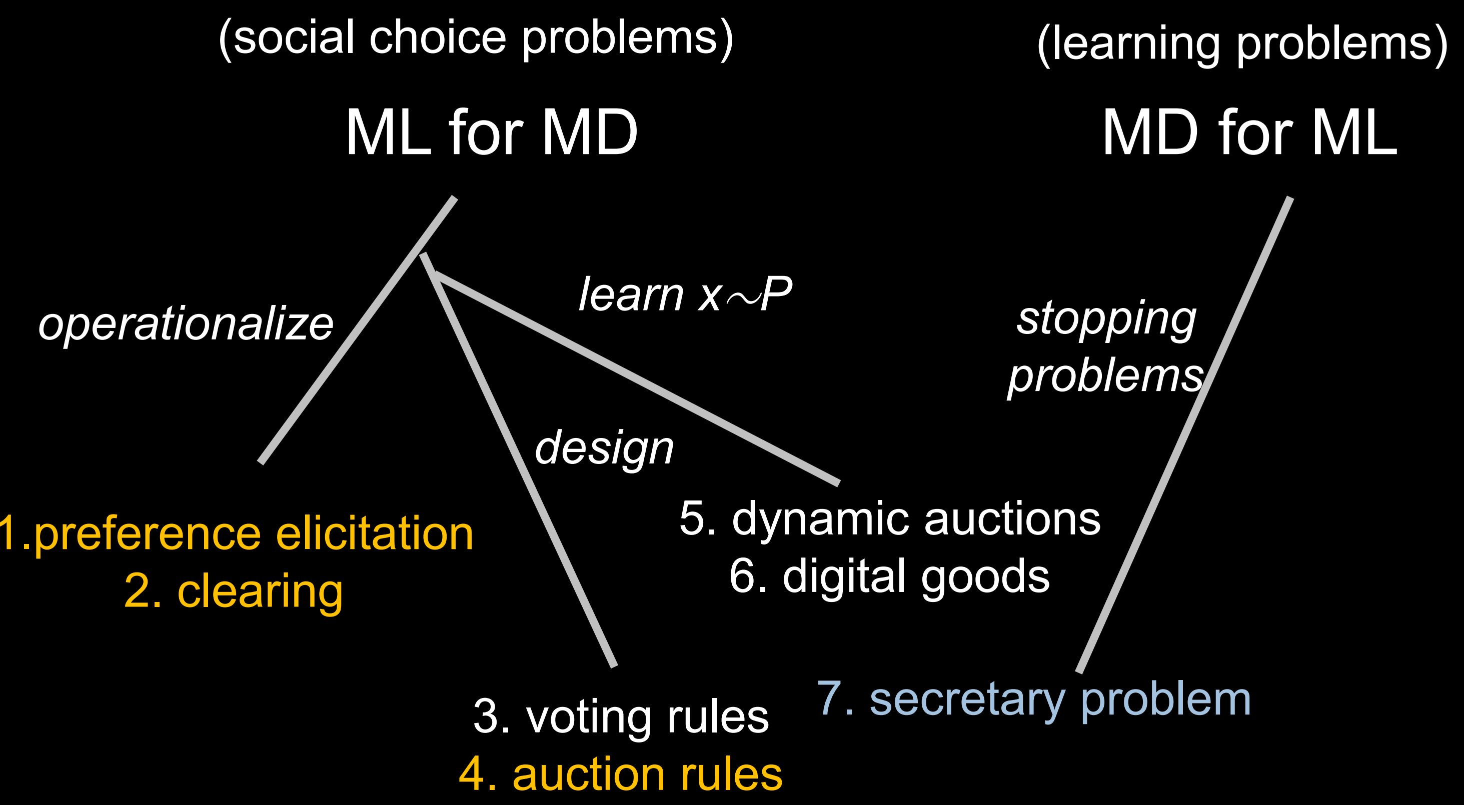
(Duetting, Fischer, Jirapinyo, Lai, Lubin and P.'10)

- $x \in R^{2^m \times n}$; $y \in \{0,1\}^m$
- Learn h: $X \to Y$; $h(x) = \arg\max_y f(x,y)$
- Stipulate $f(x,y) = x_1[y] + w^T \psi(x_{-1}, y)$
- Payment: $t_1(x,y) = - w^T \psi(x_{-1}, y)$
  - discriminant $\equiv$ payoff
- Structural SVM
- Training = minimize regularized upper bound on empirical regret.

General problem

(Duetting, Fischer, Jirapinyo, Lai, Lubin and P.'10)

- $x \in R^{2^m \times n}$; $y \in \{0,1\}^m$
- Learn h: $X \rightarrow Y$; $h(x) = \arg\max_y f(x,y)$
- Stipulate $f(x,y) = x_1[y] + w^T \psi(x_{-1}, y)$
- Payment: $t_1(x,y) = - w^T \psi(x_{-1}, y)$
  - discriminant $\equiv$ payoff
- Structural SVM
- Training = minimize regularized upper bound on empirical regret.
- **Thm**. Exact classifier $\Rightarrow$ SP auction


(social choice problems)
ML for MD
(learning problems)
MD for ML
operationalize
learn x∼P
design
stopping problems
1.preference elicitation
2. clearing
5. dynamic auctions
6. digital goods
3. voting rules
4. auction rules
7. secretary problem

7. Secretary problem

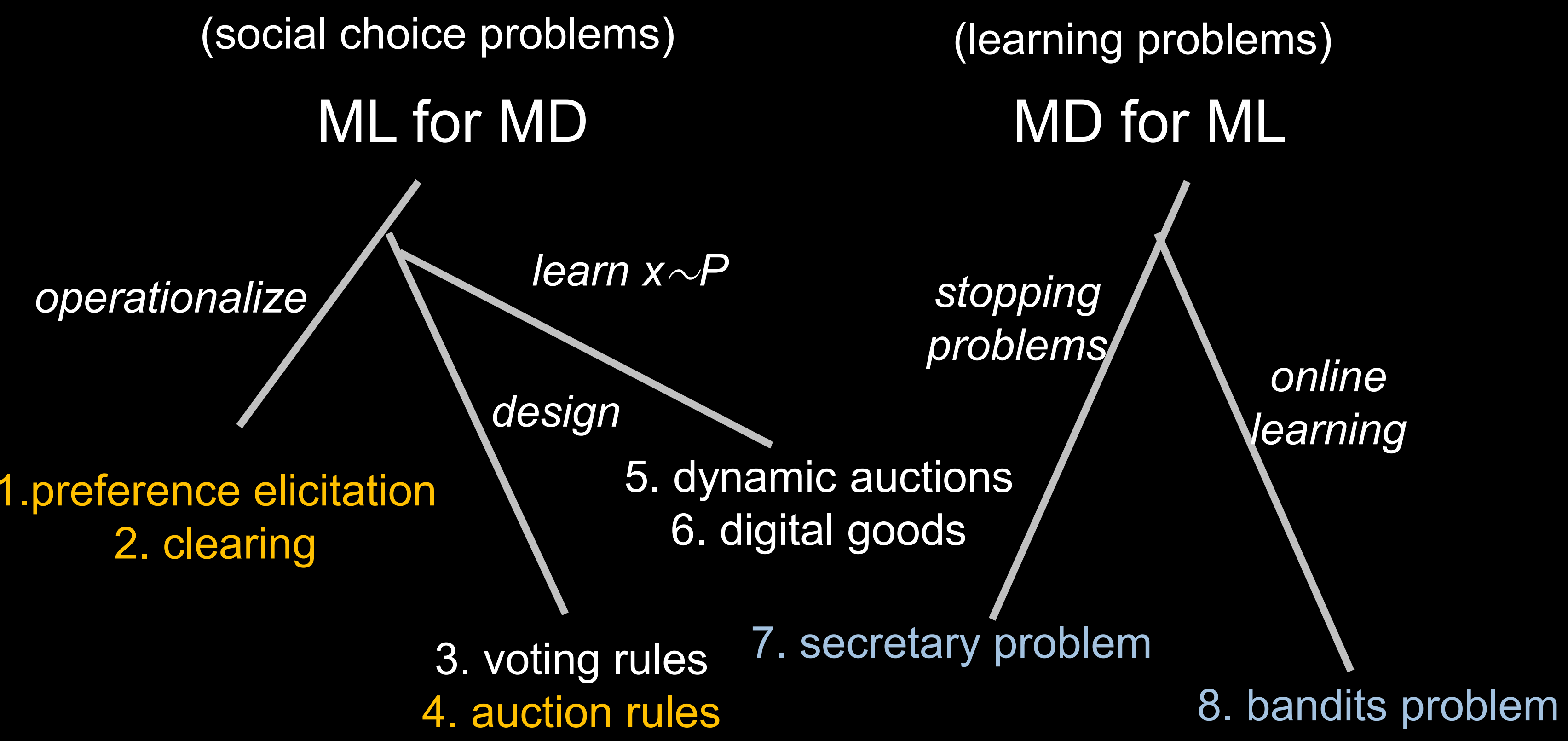
(Kleinberg, Mahdian & P. EC'03)

- Bids = secretaries
- Q: how to make the "1/e" online algorithm DSIC despite strategic inputs?

7. Secretary problem

(Kleinberg, Mahdian & P. EC'03)

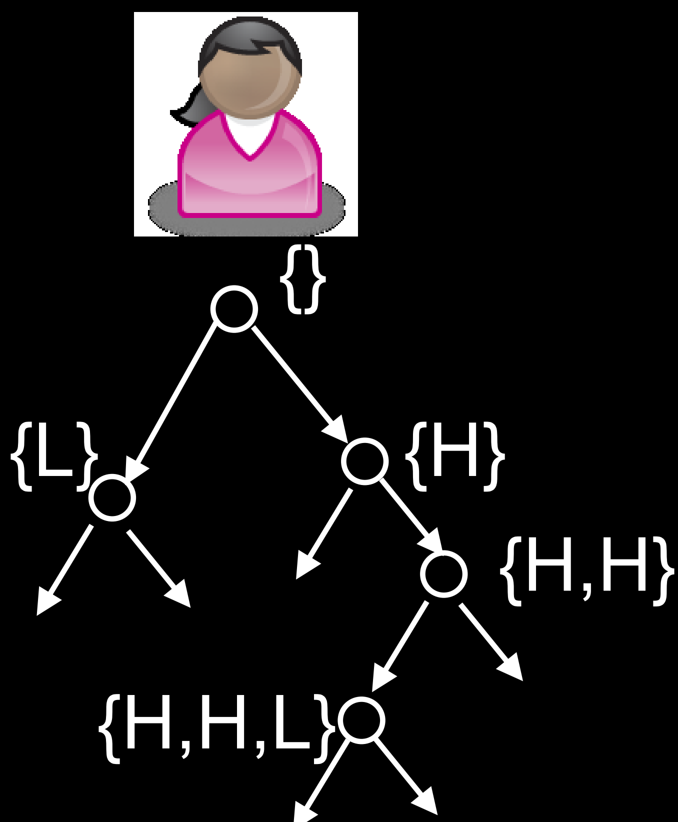
- Bids = secretaries
- Q: how to make the "1/e" online algorithm DSIC despite strategic inputs?
- A: careful handling of transition from learning to accepting.

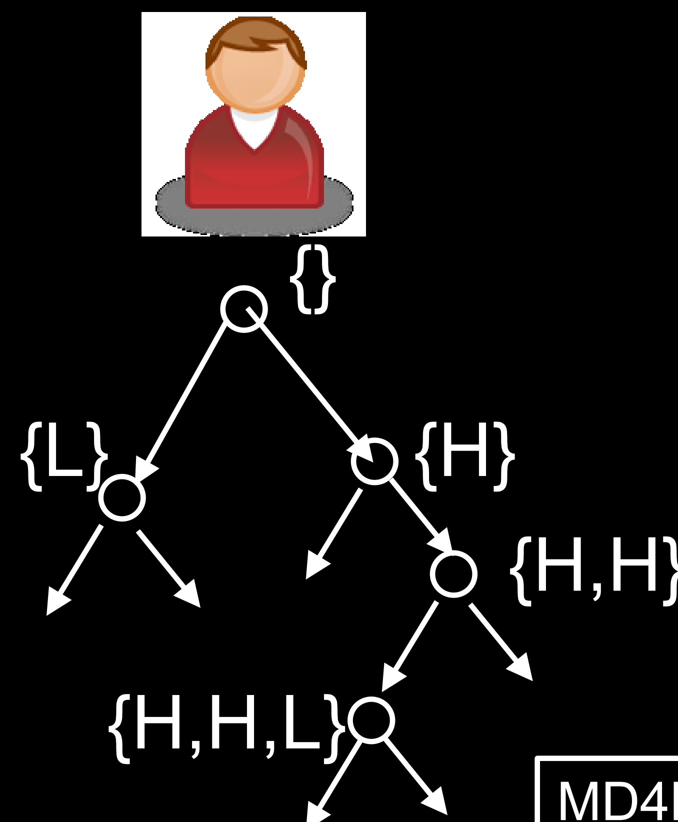

(social choice problems)
ML for MD
operationalize
design
learn x∼P
1.preference elicitation
2. clearing
3. voting rules
4. auction rules
5. dynamic auctions
6. digital goods
(learning problems)
MD for ML
stopping problems
online learning
7. secretary problem
8. bandits problem

8. Bandits problem (I)

(Cavallo, Singh & P. UAI'06', Bergemann & Valimaki'10)

- Bandits: arms = agents
- Q: how to make the agents report true reward and thus next state?

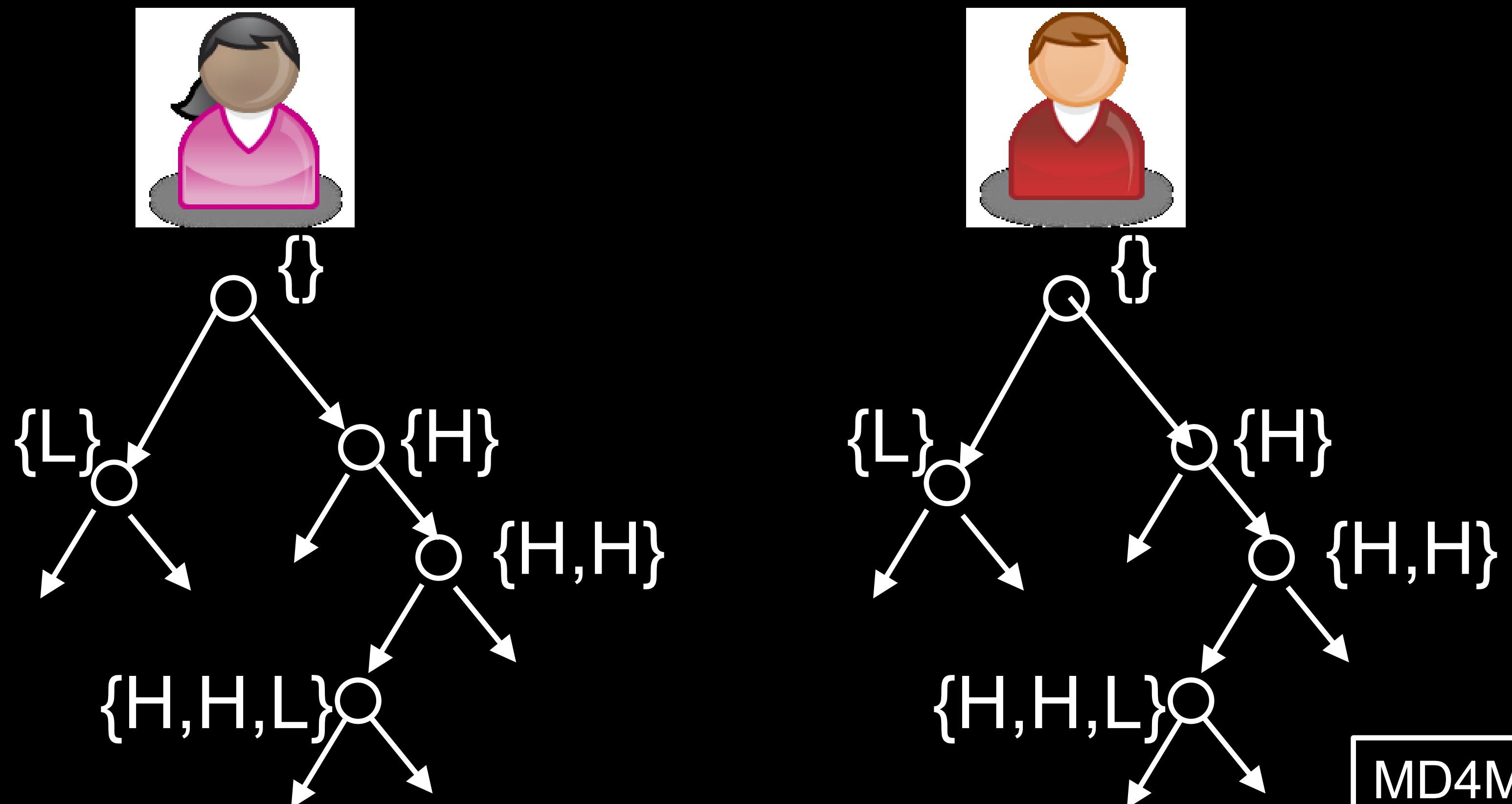




8. Bandits problem (I)

(Cavallo, Singh & P. UAI'06', Bergemann & Valimaki'10)

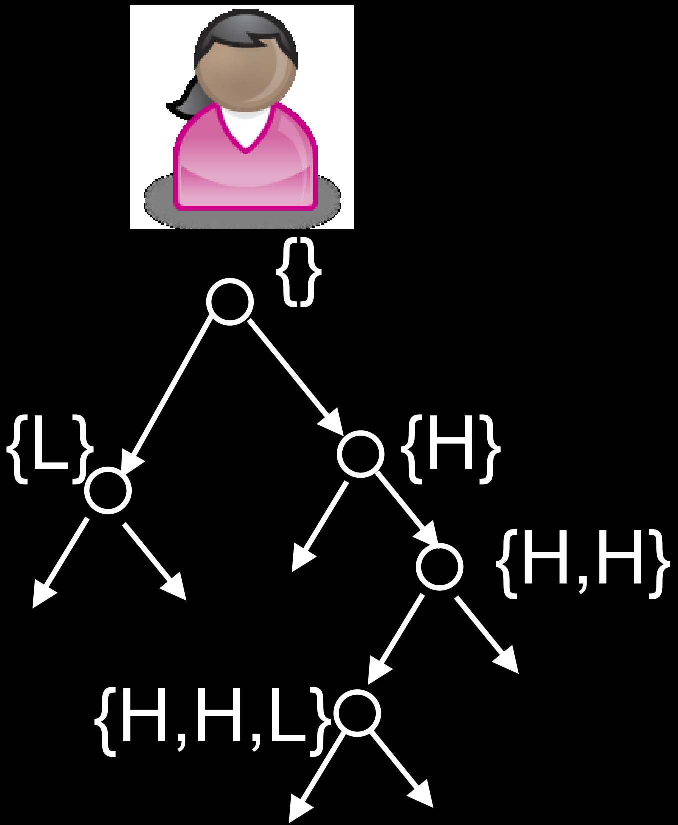
- Bandits: arms = agents
- Q: how to make the agents report true reward and thus next state? A: Dynamic VCG. (Bayesian IC)

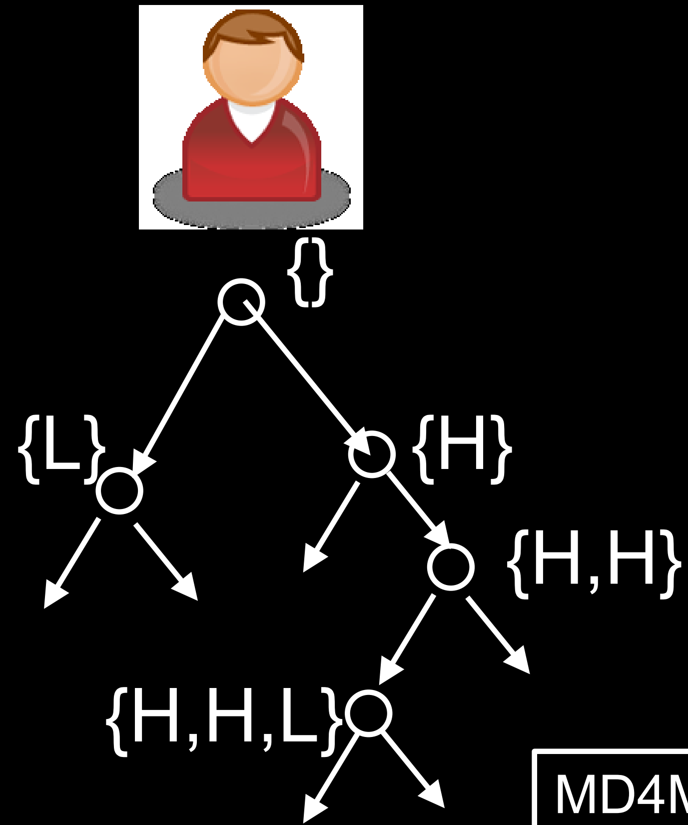


MD4ML:8

8. Bandits problem (II) (Babaioff et al. EC'09, EC'10)

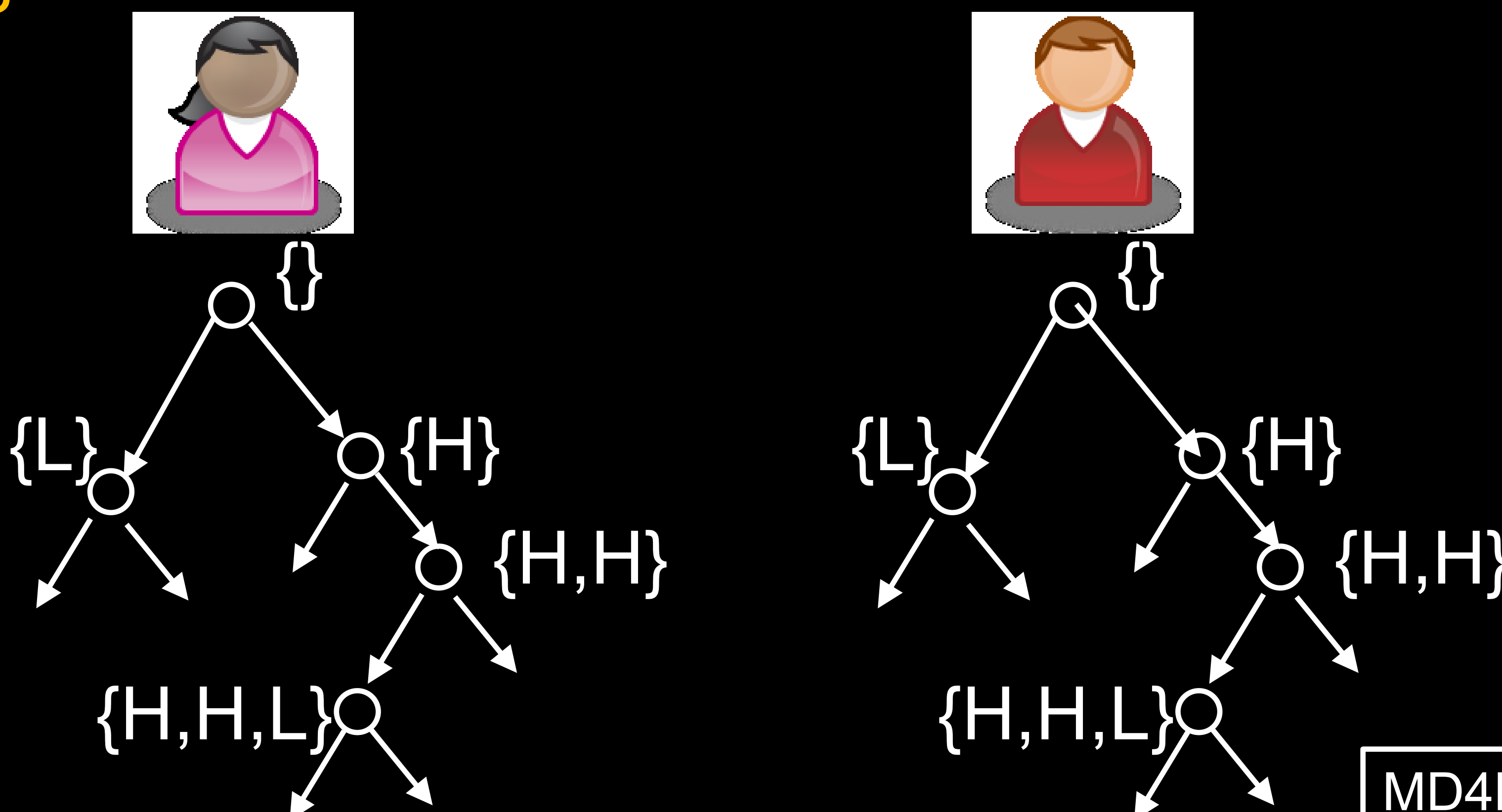
- Declare value once. Success/failure **observable**. Q: how to achieve *ex post* SP?

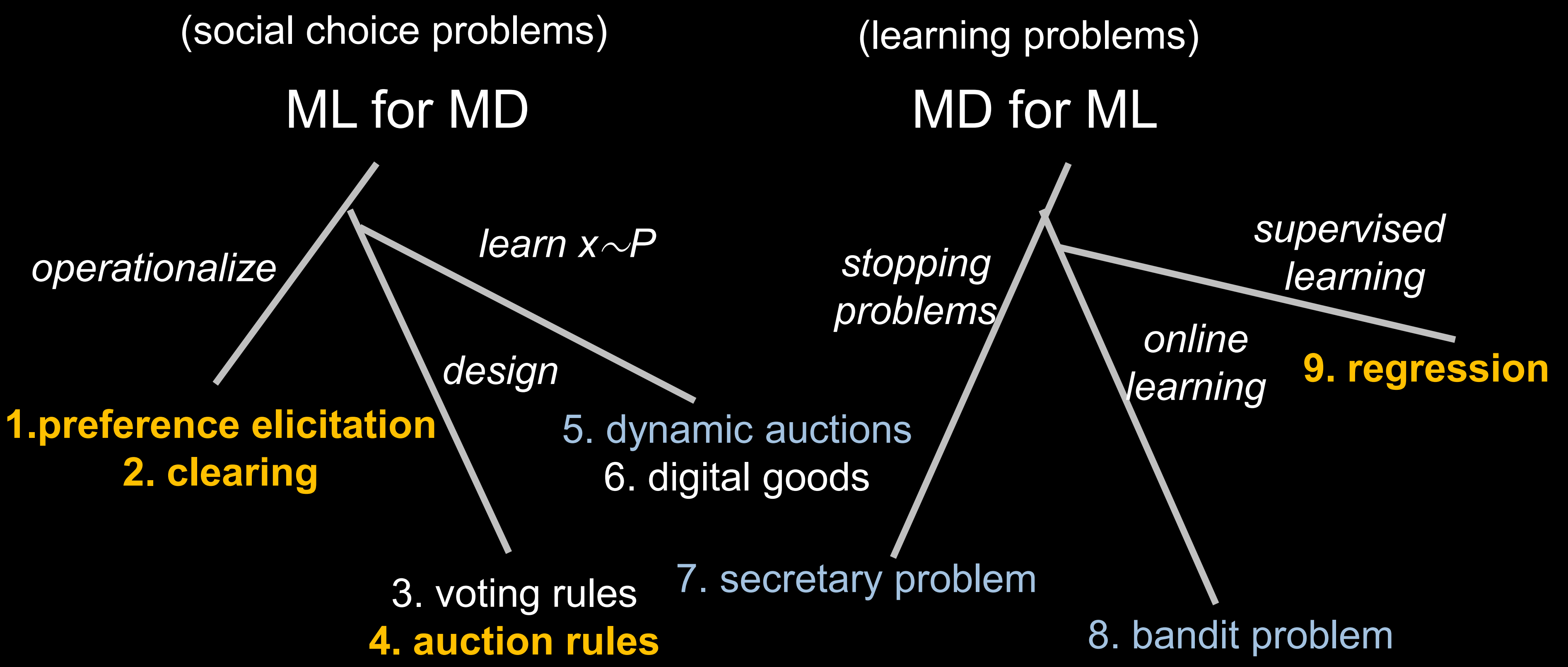




8. Bandits problem (II) (Babaioff et al. EC'09, EC'10)

- Declare value once. Success/failure **observable**. Q: how to achieve *ex post* SP?
- Deterministic: separate explor. from exploit., bad regret bound ☹. Good news for randomized ☺.




(social choice problems)
ML for MD
operationalize
learn x∼P
design
1.preference elicitation
2. clearing
5. dynamic auctions
6. digital goods
3. voting rules
4. auction rules
(learning problems)
MD for ML
stopping problems
supervised learning
online learning
9. regression
7. secretary problem
8. bandit problem

9. Incentive Compatible Regression

- Learn $f: X \to R$
- Each agent i: $x \sim P_i$; target function $g_i: X \to R$
- $R_i(f) = E_{x \sim P_i}$ [*error*($f(x), g_i(x)$)]
- Goal: $\min_f \sum_i R_i(f)$

9. Incentive Compatible Regression

- Learn f: $X \to R$
- Each agent i: $x \sim P_i$; target function g_i: $X \to R$
- $R_i(f) = E_{x \sim P_i}$ [*error*(f(x), $g_i(x)$)]
- Goal: $\min_f \sum_i R_i(f)$

- Rational agents, private knowledge of labeled examples. May misreport!

ICML: Framework

(Dekel, Fischer & Procaccia'08)

- No payments.
- Request m points $S_i = \{ (x_{ij}, y_{ij}) \}_{j=1}^{m}$
- Report $S'_i \neq S_i$. Train. Determine f.

ICML: Framework

(Dekel, Fischer & Procaccia'08)

- No payments.
- Request m points $S_i = \{ (x_{ij}, y_{ij}) \}_{j=1}^{m}$
- Report $S'_i \neq S_i$. Train. Determine f.

- One idea: select f' to be empirical risk minimizer
- Q: when will this be DSIC?

Warm-up: Special case

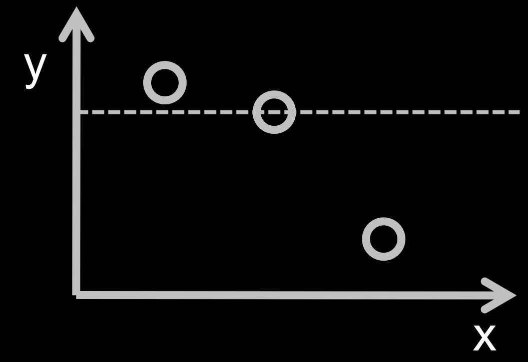
- m=1, P_i degenerate. Report y'_i

Warm-up: Special case

- m=1, P_i degenerate. Report y'_i
- Thm. For a linear |y-y'| loss function, convex hypothesis class F, then ERM is DSIC

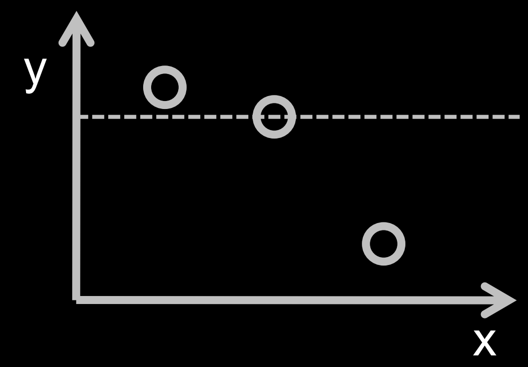
Warm-up: Special case

- m=1, P_i degenerate. Report y'_i
- Thm. For a linear |y-y'| loss function, convex hypothesis class F, then ERM is DSIC
- E.g., constant f(x)=c. ERM: select median. DSIC! 3 agent example:



Warm-up: Special case

- m=1, P_i degenerate. Report y'_i
- Thm. For a linear |y-y'| loss function, convex hypothesis class F, then ERM is DSIC
- E.g., constant f(x)=c. ERM: select median. DSIC! 3 agent example:



- Fails for squared loss $|y-y'|^2$
- E.g., {(1,2), (2,0)}.

General Case

(Dekel, Fischer & Procaccia'08)

- For m>1 points, and absolute loss function

General Case

(Dekel, Fischer & Procaccia'08)

- For m>1 points, and absolute loss function
- Example. n=2. constant $f(x)=c$
- S_1={ (1,1), (2,1), ~~(3,0)~~ } (3,1)
- S_2 = { (4,0), (5,0), (6,1) }
- $f(x)=0$; $R_1(f) = 2/3$ ⟶ $f(x)=1$; $R_1(f)=1/3$

General Case

(Dekel, Fischer & Procaccia'08)

- For m>1 points, and absolute loss function
- Example. n=2. constant f(x)=c
- S_1={ (1,1), (2,1), ~~(3,0)~~ } (3,1)
- S_2 = { (4,0), (5,0), (6,1) }
- f(x)=0; R_1(f) = 2/3 ⟶ f(x)=1; R_1(f)=1/3
- Solution: project and fit.

General Case

(Dekel, Fischer & Procaccia'08)

- For m>1 points, and absolute loss function
- Example. n=2. constant $f(x)=c$
- S_1={ (1,1), (2,1), ~~(3,0)~~ } (3,1)
- S_2 = { (4,0), (5,0), (6,1) }
- $f(x)=0$; $R_1(f) = 2/3$ ⟶ $f(x)=1$; $R_1(f)=1/3$

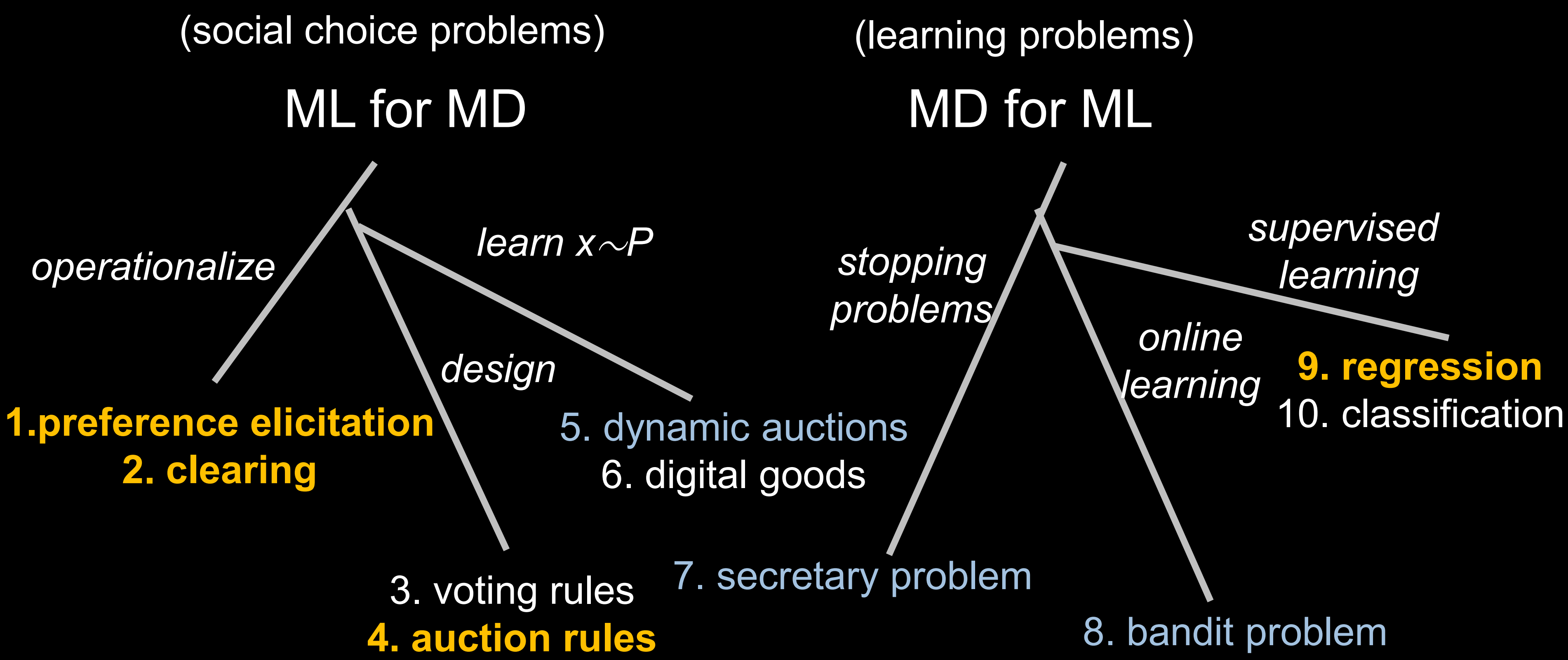
- Solution: project and fit. 3-competitive. ϵ-DSIC w/ sampling. Matching lower bound.

Challenge problem

(w/ Satinder Singh)

- Mechanism design is essentially about the design of transfer payments
- Can it be used for the design of modular intelligent systems?
  - e.g., the design of intrinsic rewards
  - e.g., the transfer of reward via payments

$\Rightarrow$ use of MD for AI/ML architectures. A "market of minds"?


(social choice problems)
ML for MD
operationalize
learn x∼P
design
1.preference elicitation
2. clearing
3. voting rules
4. auction rules
5. dynamic auctions
6. digital goods
(learning problems)
MD for ML
stopping problems
supervised learning
online learning
7. secretary problem
8. bandit problem
9. regression
10. classification

Thank you!