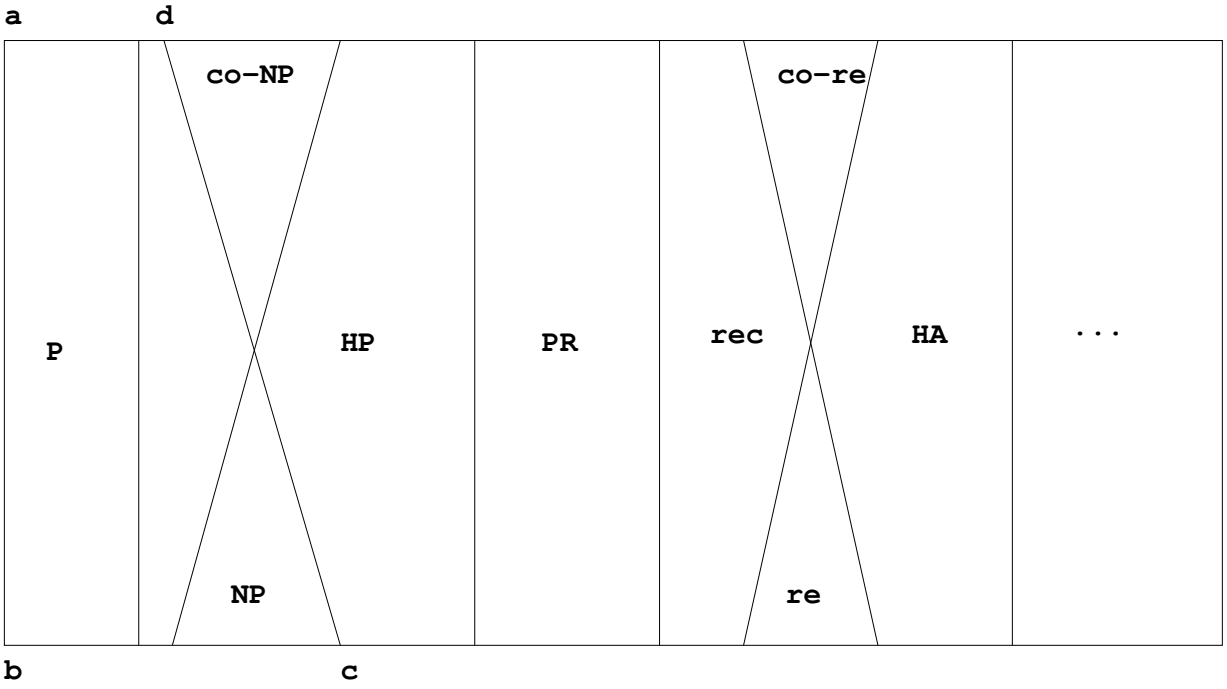
Panorama das classes de complexidade; classes aleatorizadas

Cada classe inclui todas as classes que estão para a esquerda; por exemplo, a classe **NP** corresponde ao interior do trapézio `a-b-c-d`. De forma implícita, o diagrama anterior define também as diferenças entre classes. A classe dos predicados decidíveis é a classe `rec` (predicados recursivos), a qual inclui evidentemente todos as classes à esquerda.



P : Classe dos predicados decidíveis em tempo polinomial. **NP** : Classe dos predicados decidíveis em tempo polinomial por uma máquina de Turing não determinística. `HP`: Hierarquia polinomial, $\bigcup_{i\geq 0}(\Sigma_i^p \cup \Pi_i^p)$. `PR`: Classe dos predicados definíveis por recursão primitiva. `rec`: Classe dos predicados recursivos. `re`: Classe dos predicados recursivivamente enumeráveis. `co-re`: Classe complementar de `re`. `HA`: Hierarquia aritmética, $\bigcup_{i\geq 0}(\Sigma_i \cup \Pi_i)$

Este diagrama está ainda muito incompleto...

Classes de complexidade aleatorizadas

Representamos por x os dados do problema e por y uma variável aleatória que é a testemunha do algoritmo aleatorizado ou do problema **NP**. Seja $n = |x|$ e $l(n)$ uma função de n majorada por um polinómio, sendo $n = |x|$. Quando escrevemos $y \in_u \{0,1\}^{l(n)}$ pretendemos dizer que y tem a distribuição uniforme (isto é, a mesma probabilidade, igual a $1/2^{l(n)}$) em $\{0,1\}^{l(n)}$.

Definição.

- **P** : Classe das linguagens L cuja função característica $f\colon \{0,1\}^n \to \{0,1\}$ é computável em tempo polinomial.

- **NP** : Classe das linguagens L tais que existe uma função computável em tempo polinomial $f\colon \{0,1\}^n \times \{0,1\}^{l(n)} \to \{0,1\}$ tal que

$$\begin{cases} x \in L & \Rightarrow \quad \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] > 0 \\ x \notin L & \Rightarrow \quad \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] = 0 \end{cases}$$

- **RP** : Classe das linguagens L para as quais existe uma constante $\varepsilon > 0$ e uma função computável em tempo polinomial $f\colon \{0,1\}^n \times \{0,1\}^{l(n)} \to \{0,1\}$ tais que

$$\begin{cases} x \in L & \Rightarrow \quad \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] \geq \varepsilon \\ x \notin L & \Rightarrow \quad \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] = 0 \end{cases}$$

- **BPP** : Classe das linguagens L para as quais existem constantes $\varepsilon > 0$ e ε' com $0 \leq \varepsilon < 1/2 < \varepsilon' \leq 1$ e uma função computável em tempo polinomial $f\colon \{0,1\}^n \times \{0,1\}^{l(n)} \to \{0,1\}$ tais que

$$\begin{cases} x \in L & \Rightarrow \quad \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] \geq \varepsilon' \\ x \notin L & \Rightarrow \quad \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] \leq \varepsilon \end{cases}$$

Exercício 1 Mostre que esta definição da classe **NP** coincide com a definição que conhece.

Exercício 2 Se conhecer (apenas) o algoritmo aleatorizado de Rabin-Miller para a primalidade, o que pode dizer sobre a classe do problema da primalidade (ou da linguagem correspondente)? E, na realidade, a que classe pertence?

Exercício 3 Mostre que

a) **RP**$\subseteq$**NP**, co-**RP**$\subseteq$co-**NP**.

b) co-**BPP**=**BPP**

Exercício 4 Mostre que poderíamos ter arbitrado $\varepsilon = 1/2$ na definição de **RP**. A solução deste problema usa um algoritmo que consiste executar um número fixo um outro algoritmo.

Exercício 5 Mostre que poderíamos ter arbitrado $\varepsilon = 1/3$ e $\varepsilon' = 2/3$ na definição de **BPP**.

Da aula anterior

Exercício 6 Mostre que o algoritmo de Rabin-Miller pode ser implementado em tempo polinomial (em termos de $\log n$).

Exercício 7 Suponha que pretende saber se um inteiro dado é primo ou não com uma probabilidade de erro inferior a 0.000 000 001. Quantas vezes deve correr o algoritmo anterior para garantir esse majorante do erro? Se obtiver o resultado "`COMPOSTO`", que pode concluir? E se obtiver o resultado "`PRIMO`"?

Da aula seguinte

Exercício 8 *Elemento de ordem fixa*
Seja v um vector com n elementos. Mostre que, para qualquer inteiro não negativo fixo i é possível em tempo $O(n)$ com um algoritmo simples obter o elemento de ordem i do vector v. O que pode afirmar sobre o coeficiente de n relativo ao tempo de execução do algoritmo que descreveu?