PSPACE, NPSPACE, Savitch's Theorem

True Quantified Boolean Formulas (TQBF)

$$\forall w\ \exists x\ \forall y\ \exists z\ (w \vee x \vee \neg y) \wedge (\neg w \vee \neg x) \wedge (x \vee y \vee \neg z) \wedge z$$

- **Instance:** Given a quantified Boolean formula (QBF) ϕ
- **Problem:** Is ϕ true?

True Quantified Boolean Formulas (TQBF)

$$\forall w\ \exists x\ \forall y\ \exists z\ (w \vee x \vee \neg y) \wedge (\neg w \vee \neg x) \wedge (x \vee y \vee \neg z) \wedge z$$

- **Instance:** Given a quantified Boolean formula (QBF) ϕ
- **Problem:** Is ϕ true?

- Find a simple algorithm for TQBF.
- What is its time complexity?

True Quantified Boolean Formulas (TQBF)

$$\forall w\ \exists x\ \forall y\ \exists z\ (w \vee x \vee \neg y) \wedge (\neg w \vee \neg x) \wedge (x \vee y \vee \neg z) \wedge z$$

- **Instance:** Given a quantified Boolean formula (QBF) ϕ
- **Problem:** Is ϕ true?

- Might TQBF be complete for EXP?
- Compare with Generalized Checkers (GC)
  - Both problems are game-like
  - Both can be modeled by a graph with exponentially many nodes
  - But (unlike GC), the TQBF graph is a tree of polynomial depth
- *TQBF has an algorithm that uses polynomial space*

Space Bounded Complexity Classes

Space Bounded Complexity Classes

- Distinguish between a *read-only input tape* and *work tapes* of a Turing Machine (TM).
- SPACE($s(n)$) is the set of languages accepted by deterministic TMs that always halt and use $O(s(n))$ work tape cells on inputs of length n.
- NSPACE($s(n)$): replace "deterministic" by "nondeterministic". (Assume that regardless of nondeterministic choices made, the TM halts.)
- $s(n) \geqslant \log n$ is *space-constructible* if there is a DTM that on input 1^n computes $1^{s(n)}$ in $O(s(n))$ space.

Space Bounded Complexity Classes

- PSPACE = $\bigcup_{c>0}$ SPACE(n^c)
- NPSPACE = $\bigcup_{c>0}$ NSPACE(n^c)

Space Bounded Complexity Classes

- PSPACE = $\bigcup_{c>0}$ SPACE(n^c)
- NPSPACE = $\bigcup_{c>0}$ NSPACE(n^c)

- Explain why PSPACE $\subseteq$ EXP

Space Bounded Complexity Classes

- PSPACE = $\bigcup_{c>0}$ SPACE(n^c)
- NPSPACE = $\bigcup_{c>0}$ NSPACE(n^c)

- Explain why PSPACE $\subseteq$ EXP
  - A TM using O(n^c) space has $2^{O(n^c)}$ configurations on an input of length n.
  - If the TM halts, none can be visited more than once on any computation. So the TM uses time $2^{O(n^c)}$.

Space Bounded Complexity Classes

- PSPACE = $\bigcup_{c>0}$ SPACE(n^c)
- NPSPACE = $\bigcup_{c>0}$ NSPACE(n^c)

- Explain why PSPACE $\subseteq$ EXP

Space Bounded Complexity Classes

- PSPACE = $\bigcup_{c>0}$ SPACE(n^c)
- NPSPACE = $\bigcup_{c>0}$ NSPACE(n^c)

- Explain why PSPACE $\subseteq$ EXP
- Explain why NP $\subseteq$ PSPACE

Space Bounded Complexity Classes

- PSPACE = $\cup_{c>0}$ SPACE(n^c)
- NPSPACE = $\cup_{c>0}$ NSPACE(n^c)

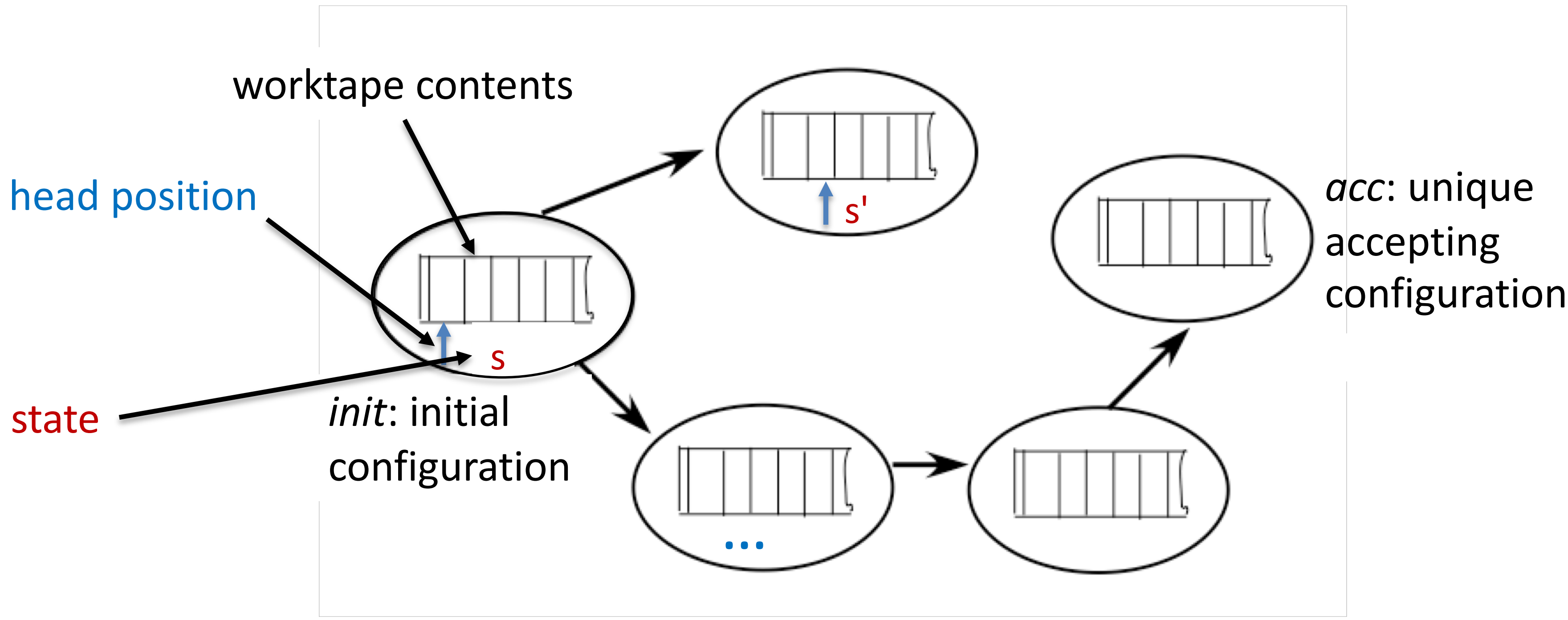
- Explain why PSPACE $\subseteq$ EXP
- Explain why NP $\subseteq$ PSPACE
- What about co-NP?

Space Bounded Complexity Classes

- PSPACE = $\bigcup_{c>0}$ SPACE(n^c)
- NPSPACE = $\bigcup_{c>0}$ NSPACE(n^c)

- What about NPSPACE?

Configuration Graph of NTM M on input *w*



- Nodes represent *configurations* (state, head position, work tape contents) of M on *w*
- Edges represent transitions
- Since M always halts, the graph is acyclic
- If M is $s(n)$-space bounded, can you bound the number of nodes of the graph as a function of $|w|$?

NPSPACE $\subseteq$ EXP

- Let M be a NTM using O(n^c) space .
- Exp-time algorithm for L(M): On input w:
  – Write down the configuration graph of M on w; size of the graph is $2^{O(|w|^c)}$
  – Check if the accepting configuration can be reached from the initial configuration (use depth first search or breadth first search)

NPSPACE $\subseteq$ EXP

- Let M be a NTM using $O(n^c)$ space .
- Exp-time algorithm for L(M): On input w:
  - Write down the configuration graph of M on w; size of the graph is $2^{O(|w|^c)}$
  - Check if the accepting configuration can be reached from the initial configuration (use depth first search or breadth first search)
- More generally, if $s(n)$ is space-constructible then

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DTIME}(2^{O(s(n))})$$

Savitch's Theorem: NSPACE($s(n)$) $\subseteq$ DSPACE($s(n)^2$)

Savitch's Theorem: NSPACE($s(n)$) $\subseteq$ DSPACE($s(n)^2$)

- Proof idea: Let L be accepted by NTM M within $c.s(n)$) space and $2^{c.s(n)}$ time. We'll describe a deterministic algorithm that accepts L in O($s(n)^2$) space.
- Fix input *w*, let G be the configuration graph of M on *w*.

Savitch's Theorem: NSPACE($s(n)$) ⊆ DSPACE($s(n)^2$)

- Proof idea: Let L be accepted by NTM M within $c.s(n)$) space and $2^{c.s(n)}$ time. We'll describe a deterministic algorithm that accepts L in $O(s(n)^2)$ space.
- Fix input *w,* let G be the configuration graph of M on *w.*
- Let Reach(x,y,i) be true if there is a path of length $\leq 2^i$ from node x to node y in G, and false otherwise.
- On input *w*, compute

$$\text{Reach}(init, acc, c.s(|w|))$$

and accept if and only if the function returns true

Savitch's Theorem: NSPACE($s(n)$) $\subseteq$ DSPACE($s(n)^2$)

Reach(x,y,i) // does G have a path of length $\leq 2^i$ from x to y?

Savitch's Theorem: NSPACE($s(n)$) $\subseteq$ DSPACE($s(n)^2$)

Reach(x,y,i) // does G have a path of length $\leq 2^i$ from x to y?

  If $i = 0$ then

    If ($x = y$) or ((x,y) is an edge of G) Return True

    Else Return False

  Else

    For each node z of G

      If (Reach (x, z, i-1) and Reach(z, y, i-1))

        Return True

    Return False

Savitch's Theorem: NSPACE($s(n)$) $\subseteq$ DSPACE($s(n)^2$)

- The space per recursion level is proportional to the space, s(|*w*|), used by M on *w.*
- The recursion depth is *i*
- So, the recursion depth is *c*.s(|*w*|) on call

  Reach(*init*, *acc*, *c*.s(|*w*|)),

  and the total space used is $O(s(|w|)^2$.

Savitch's Theorem: NSPACE($s(n)$) $\subseteq$ DSPACE($s(n)^2$)

- The space per recursion level is proportional to the space, s(|*w*|), used by M on *w.*
- The recursion depth is *i*
- So, the recursion depth is *c*.s(|*w*|) on call

  Reach(*init*, *acc*, *c*.s(|*w*|)),

  and the total space used is $O(s(|w|)^2$.

- Note: Space constructability is useful for the proof, to write down the bound *c*.s(|*w*|). However, the algorithm could be re-run with successively larger bounds until a sufficiently large one is found.

TQBF is PSPACE-complete

TQBF is PSPACE-complete

- Let L be a PSPACE language, accepted by TM M within space $c.s(n)$ and time $2^{c.s(n)}$.

- Goal: Poly-time reduction $w \rightarrow$ QBF(w) such that

  w is in L iff QBF(w) is true.

- Equivalently, if Reach(x,y,i) is as before, then

  Reach($init$,acc,c.s($|w|$)) iff QBF(w) is true.

TQBF is PSPACE-complete

Reach(x,y,i) // does G have a path of length ≤ 2^i from x to y?

If i = 0 [base case omitted]

Else

For each node z of G

If (Reach(x, z, i-1) and Reach(z, y, i-1))

Return True

Return False

First try at expressing this using logic:

$$\exists \text{ config } z\text{: Reach}(x,z,i\text{-}1) \wedge \text{Reach}(z,y,i\text{-}1)\ )$$

TQBF is PSPACE-complete

Reach(x,y,i) // does G have a path of length $\leq 2^i$ from x to y?

    If $i = 0$ [base case omitted]

    Else

        For each node z of G

            If (Reach(x, z, i-1) and Reach(z, y, i-1))

                Return True

        Return False

First try at expressing this using logic:

$$\exists \text{ config } z\text{: Reach}(x,z,i\text{-}1) \land \text{Reach}(z,y,i\text{-}1))$$

Problem: formula size will blow up when expanding the Reach expressions, because of doubling

TQBF is PSPACE-complete

Reach(x,y,i) // does G have a path of length ≤ 2^i from x to y?

If i = 0 [base case omitted]

Else

```
For each node z of G
    If (Reach(x, z, i-1) and Reach(z, y, i-1) )
        Return True
Return False
```

Better way of expressing this using logic:

$\exists$ config z $\forall$ $v \in \{\text{True, False}\}$ $\exists$ configs z', z''

$(v \Rightarrow z',z''=x,z) \wedge (\neg v \Rightarrow z',z''=z,y) \wedge \text{Reach}(z',z'',i\text{-}1)$

TQBF is PSPACE-complete

Overall QBF:

$$\exists\, z_1\, \forall\, v_1\, \exists\, z_1', z_1''\ \exists\, z_2\, \forall\, v_2 \ldots \exists\, z_m', z_m''$$

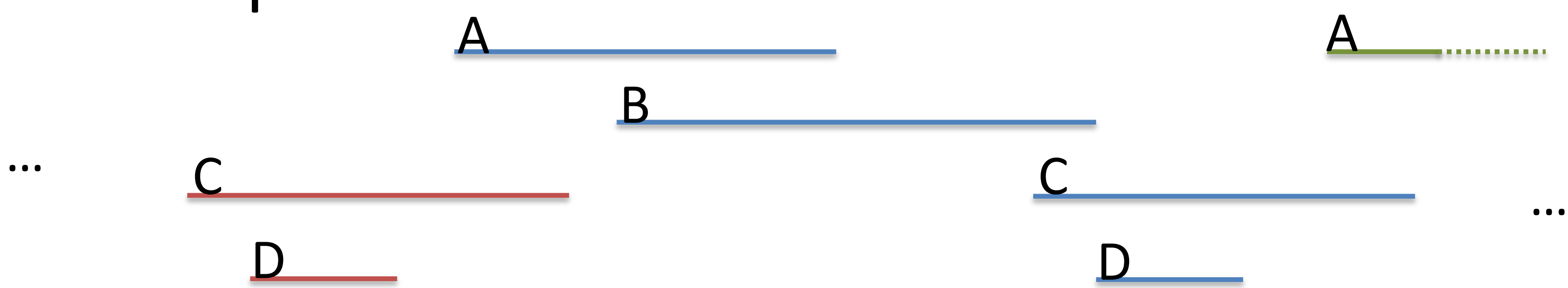
$$\phi(z_0', z_0'', z_1, v_1, z_1', z_1'', ,\ldots, z_m, v_m, z_m', z_m'')$$

where $m = c.s(n)$ and ϕ encodes that

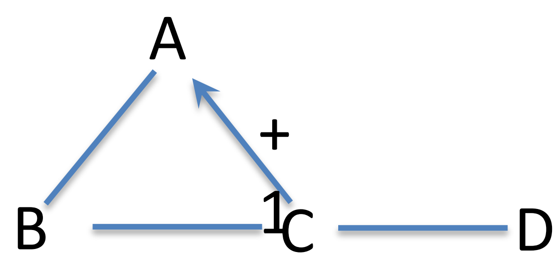
- z_0' and z_0'' are the initial and accepting configs of M on w
- for each i, if v_i = true then $z_i', z_i'' = z_{i-1}', z_i$
- for each i, if v_i = false then $z_i', z_i'' = z_i, z_{i-1}''$
- all of the z_i, z_i' and z_i'' encode valid configurations
- $z_m' = z_m''$ or (z_m', z_m'') is an edge of G (base case)

Periodic Graph Colouring

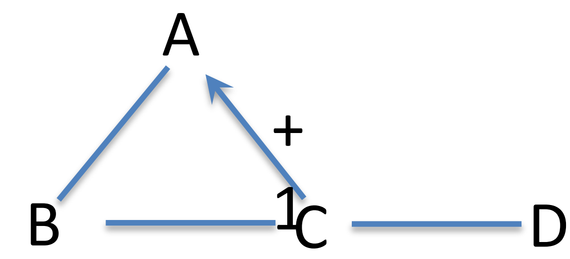
- Motivation: schedule jobs at the same time period each day; want to minimize processors
- Example:



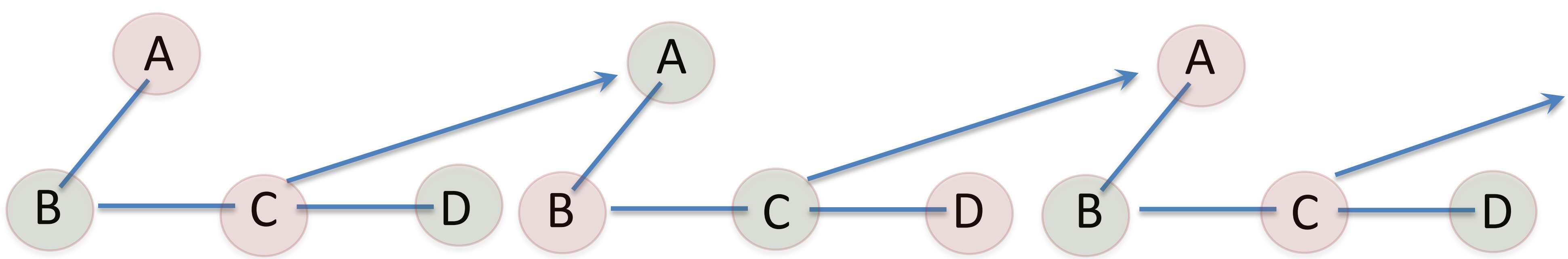
- Succinct representation:



Periodic Graph Colouring



- Succinct graph is 3-colourable, suggesting that we need 3 processors (since two jobs in overlapping time intervals cannot be scheduled on the same processor)
- But the infinite graph is actually 2-colourable, and so we can use just 2 processors!



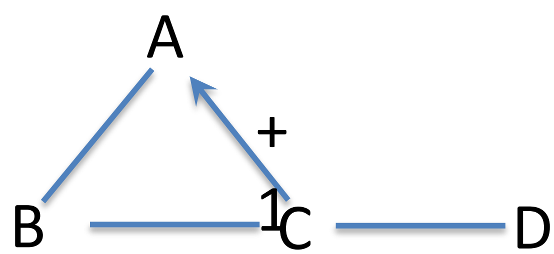
Periodic Graph Colouring

- A *periodic graph* G is an infinite undirected graph, specified by a triple (V,E,E')

- G's nodes:
  - $\bigcup V_i$ where $V_i = \{v_i \mid v \text{ in } V\}$, for all i in $\mathbb{Z}$
- G's edges: $(\bigcup E_i) \cup (\bigcup E_i')$
  - where $E_i = \{\{u_i\ v_i\} \mid \{u,v\} \text{ in } E\}$
  - and $E_i' = \{ \{u_i\ v_{i+1}\} \mid (u,v) \text{ in } E'\}$

Periodic Graph Colouring

- **Instance**: A periodic graph G = (V,E,E') and a positive number *k*
- **Problem**: Is G *k*-colourable?

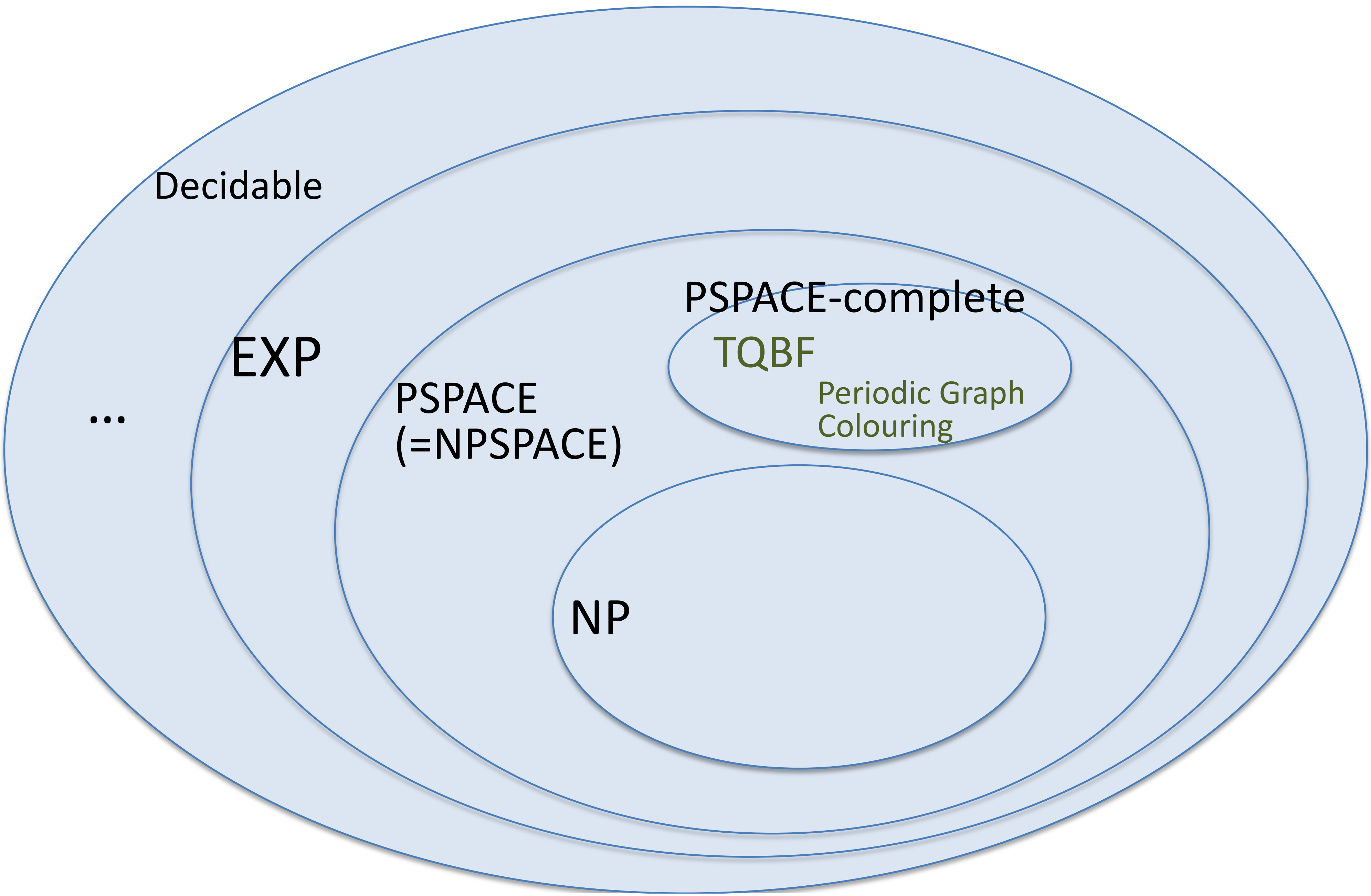


- Can you suggest a nondeterministic algorithm for Periodic Graph Colouring that runs in polynomial space?

Summary

- PSPACE refines the categorization of problems within EXP: those that can be solved with only polynomial space vs those that seem to need both exponential time *and* space
- NPSPACE = PSPACE! (Savitch's Theorem)
- We can leverage Savitch's Theorem to simplify proofs that some problems are in PSPACE (e.g., Periodic Graph Colouring, which also happens to be PSPACE-complete.

Summary


Decidable
EXP
...
PSPACE
(=NPSPACE)
PSPACE-complete
TQBF
Periodic Graph
Colouring
NP

Next Class

- Space bounded classes within P
- Arora-Barak, 4.1-4.3; 6.5