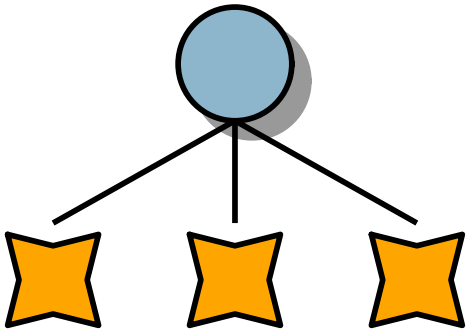
IOzSeF

Integrated Oz Search Factory



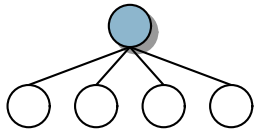
Fortgeschrittenen-Praktikum von

Guido Tack

Betreut von

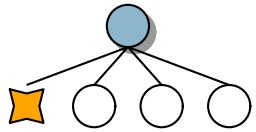
Christian Schulte

Überblick



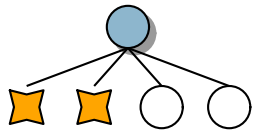
Überblick

- Was ist eine *Search Factory*?



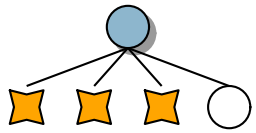
Überblick

- Was ist eine *Search Factory*?
- Orthogonale Aspekte von Suche



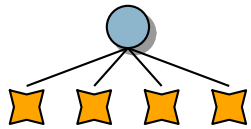
Überblick

- Was ist eine *Search Factory*?
- Orthogonale Aspekte von Suche
- Implementierung

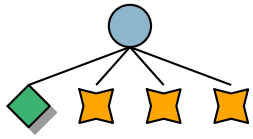


Überblick

- Was ist eine *Search Factory*?
- Orthogonale Aspekte von Suche
- Implementierung
- Evaluation

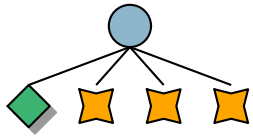


1 Was ist eine *Search Factory*?



1 Was ist eine *Search Factory*?

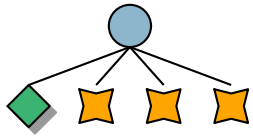
Eine *Search Factory*



1 Was ist eine *Search Factory*?

Eine *Search Factory*

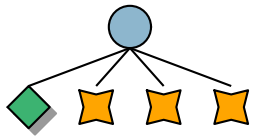
- Kombiniert orthogonale Aspekte von Suche



1 Was ist eine *Search Factory*?

Eine *Search Factory*

- Kombiniert orthogonale Aspekte von Suche
- Versteckt orthogonale Aspekte hinter einfachen Interfaces

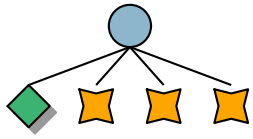


1 Was ist eine *Search Factory*?

Eine *Search Factory*

- Kombiniert orthogonale Aspekte von Suche
- Versteckt orthogonale Aspekte hinter einfachen Interfaces

Wozu?



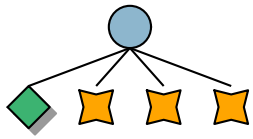
1 Was ist eine *Search Factory*?

Eine *Search Factory*

- Kombiniert orthogonale Aspekte von Suche
- Versteckt orthogonale Aspekte hinter einfachen Interfaces

Wozu?

- Suche in Mozart ist bisher wenig anpassbar



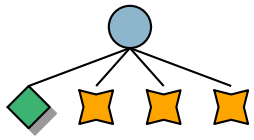
1 Was ist eine *Search Factory*?

Eine *Search Factory*

- Kombiniert orthogonale Aspekte von Suche
- Versteckt orthogonale Aspekte hinter einfachen Interfaces

Wozu?

- Suche in Mozart ist bisher wenig anpassbar
  (Feste Bibliotheksfunktionen, Explorer fest verdrahtet)



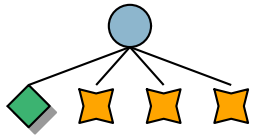
1 Was ist eine *Search Factory*?

Eine *Search Factory*

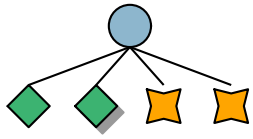
- Kombiniert orthogonale Aspekte von Suche
- Versteckt orthogonale Aspekte hinter einfachen Interfaces

Wozu?

- Suche in Mozart ist bisher wenig anpassbar
  (Feste Bibliotheksfunktionen, Explorer fest verdrahtet)
- Orthogonalität verschiedener Konzepte bisher nicht implementiert

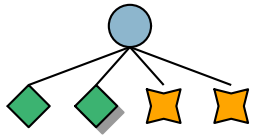


2 Orthogonale Aspekte von Suche



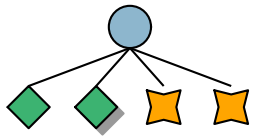
2 Orthogonale Aspekte von Suche

- Recomputation



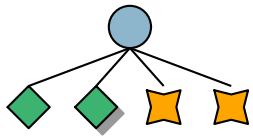
2 Orthogonale Aspekte von Suche

- Recomputation
- Explorations-Strategie



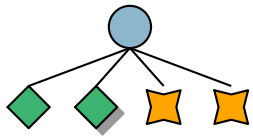
2 Orthogonale Aspekte von Suche

- Recomputation
- Explorations-Strategie
- Optimierung (Branch & Bound)



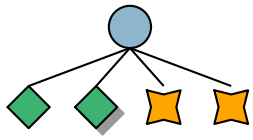
2 Orthogonale Aspekte von Suche

- Recomputation
- Explorations-Strategie
- Optimierung (Branch & Bound)
- Visualisierung / Zugriff auf Lösungen



2 Orthogonale Aspekte von Suche

- Recomputation
- Explorations-Strategie
- Optimierung (Branch & Bound)
- Visualisierung / Zugriff auf Lösungen
- Parallelisierung



Recomputation

Recomputation

- Recomputation macht Suche erst effizient

Recomputation

- Recomputation macht Suche erst effizient
- IOzSeF macht Recomputation transparent

Recomputation

- Recomputation macht Suche erst effizient
- IOzSeF macht Recomputation transparent
  (Vollständige Verwaltung der Suchbaum-Datenstruktur)

Recomputation

- Recomputation macht Suche erst effizient
- IOzSeF macht Recomputation transparent
  (Vollständige Verwaltung der Suchbaum-Datenstruktur)
- Implementierte Methoden:

Recomputation

- Recomputation macht Suche erst effizient
- IOzSeF macht Recomputation transparent
  (Vollständige Verwaltung der Suchbaum-Datenstruktur)
- Implementierte Methoden:
  - Fixed (feste Distanz, nach der Kopie erzeugt wird)

Recomputation

- Recomputation macht Suche erst effizient
- IOzSeF macht Recomputation transparent
  (Vollständige Verwaltung der Suchbaum-Datenstruktur)
- Implementierte Methoden:
  - Fixed (feste Distanz, nach der Kopie erzeugt wird)
  - Adaptiv (Distanz wird während der Exploration angepasst)

Exploration

Exploration

- Explorations-Strategien sollen vom Benutzer programmiert werden können

Exploration

- Explorations-Strategien sollen vom Benutzer programmiert werden können
- Einfaches Baum-Interface

Exploration

- Explorations-Strategien sollen vom Benutzer programmiert werden können
- Einfaches Baum-Interface
- Automatische Binarisierung
  (wichtig z.B. bei Limited Discrepancy Search)

Exploration

- Explorations-Strategien sollen vom Benutzer programmiert werden können
- Einfaches Baum-Interface
- Automatische Binarisierung
  (wichtig z.B. bei Limited Discrepancy Search)
- Exploration in Phasen möglich

Exploration

- Explorations-Strategien sollen vom Benutzer programmiert werden können
- Einfaches Baum-Interface
- Automatische Binarisierung
  (wichtig z.B. bei Limited Discrepancy Search)
- Exploration in Phasen möglich
  (LDS, Iterative Deepening)

Optimierung

Optimierung

- Branch & Bound:

Optimierung

- Branch & Bound:
  Lösung gefunden $\Rightarrow$ nachfolgend explorierte Knoten verbessern

Optimierung

- Branch & Bound:
  Lösung gefunden $\Rightarrow$ nachfolgend explorierte Knoten verbessern
- Best Solution Search (iterativ):

Optimierung

- Branch & Bound:
  Lösung gefunden $\Rightarrow$ nachfolgend explorierte Knoten verbessern
- Best Solution Search (iterativ):
  Lösung gefunden $\Rightarrow$ von vorn beginnen mit
  "Verbesserungs-Constraints"

Optimierung

- Branch & Bound:
  Lösung gefunden $\Rightarrow$ nachfolgend explorierte Knoten verbessern
- Best Solution Search (iterativ):
  Lösung gefunden $\Rightarrow$ von vorn beginnen mit
  "Verbesserungs-Constraints"
  (noch nicht implementiert)

Visualisierung

Visualisierung

- Debugging-Werkzeug

Visualisierung

- Debugging-Werkzeug
- Effizienz der Modellierung verbessern

Visualisierung

- Debugging-Werkzeug
- Effizienz der Modellierung verbessern
- Interaktive Suche / inkrementeller Aufbau

Visualisierung

- Debugging-Werkzeug
- Effizienz der Modellierung verbessern
- Interaktive Suche / inkrementeller Aufbau
- Visualisierung beliebiger Explorations-Strategien

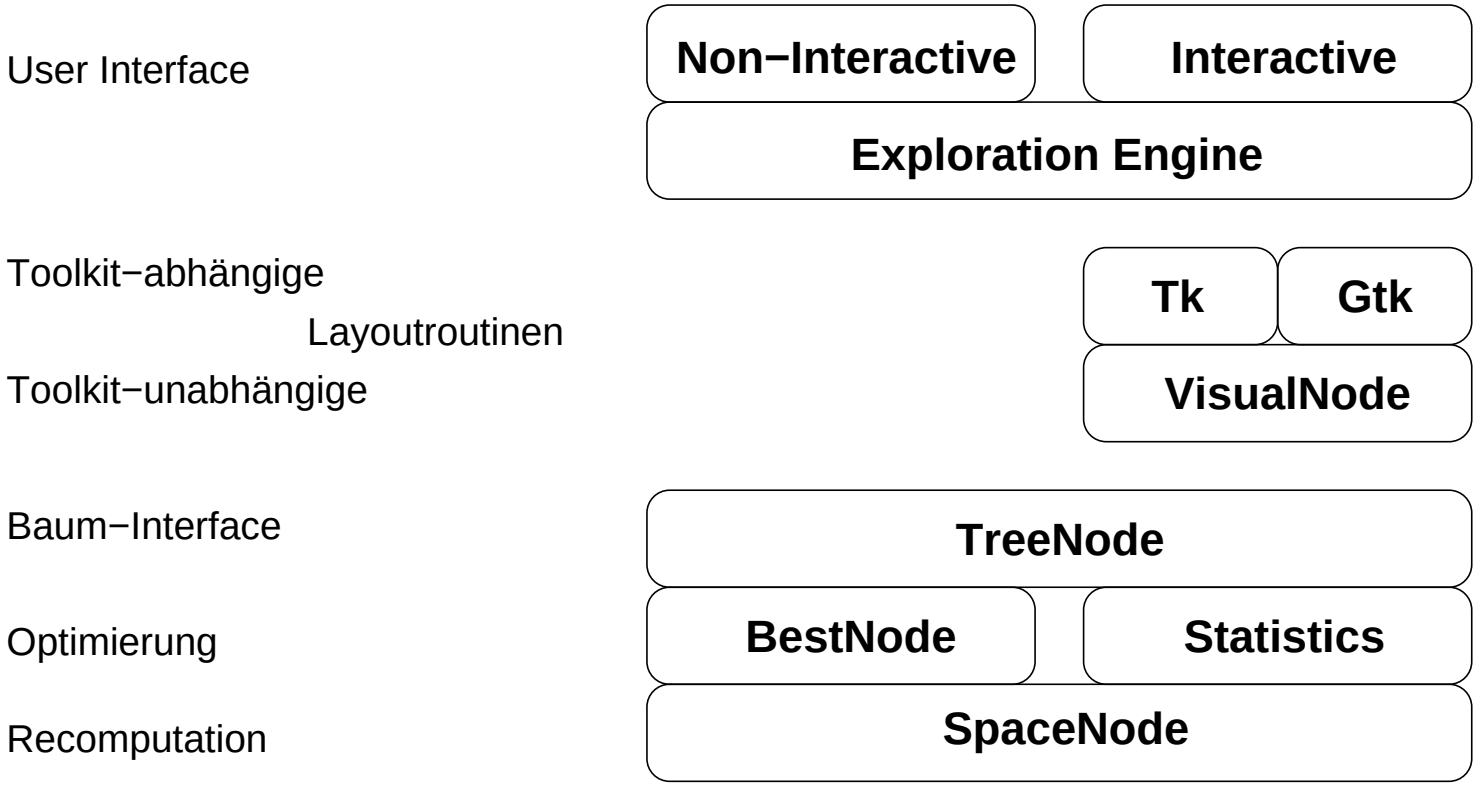
Visualisierung

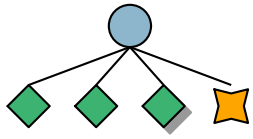
- Debugging-Werkzeug
- Effizienz der Modellierung verbessern
- Interaktive Suche / inkrementeller Aufbau
- Visualisierung beliebiger Explorations-Strategien
- Zugriff auf Lösungen von außen möglich

Visualisierung

- Debugging-Werkzeug
- Effizienz der Modellierung verbessern
- Interaktive Suche / inkrementeller Aufbau
- Visualisierung beliebiger Explorations-Strategien
- Zugriff auf Lösungen von außen möglich
- Nicht-interaktive Suche (Bibliotheksfunktionen)

3 Implementierung





Recomputation

Recomputation

- *SpaceNodes* haben evtl.

Recomputation

- *SpaceNodes* haben evtl. Arbeits-Space

Recomputation

- *SpaceNodes* haben evtl.
  Arbeits-Space und Kopie-Space

Recomputation

- *SpaceNodes* haben evtl.
  Arbeits-Space und Kopie-Space
- Folgendes Interface:
  `new(Script)`, `init(Mom I)`, `getSpace($)`, `ask($)`,
  `merge($)`

Recomputation

- *SpaceNodes* haben evtl.
  Arbeits-Space und Kopie-Space
- Folgendes Interface:
  `new(Script)`, `init(Mom I)`, `getSpace($)`, `ask($)`,
  `merge($)`
- getSpace besorgt Arbeitsspace

Recomputation

- *SpaceNodes* haben evtl.
  Arbeits-Space und Kopie-Space
- Folgendes Interface:
  `new(Script)`, `init(Mom I)`, `getSpace($)`, `ask($)`,
  `merge($)`
- getSpace besorgt Arbeitsspace
- Automatische Recomputation bei Bedarf

Recomputation

- *SpaceNodes* haben evtl.
  Arbeits-Space und Kopie-Space
- Folgendes Interface:
  `new(Script)`, `init(Mom I)`, `getSpace($)`, `ask($)`,
  `merge($)`
- getSpace besorgt Arbeitsspace
- Automatische Recomputation bei Bedarf
  (inkl. Anlegen der Kopien)

Recomputation

- *SpaceNodes* haben evtl.
  Arbeits-Space und Kopie-Space
- Folgendes Interface:
  `new(Script)`, `init(Mom I)`, `getSpace($)`, `ask($)`,
  `merge($)`
- getSpace besorgt Arbeitsspace
- Automatische Recomputation bei Bedarf
  (inkl. Anlegen der Kopien)
- Baum der *SpaceNodes* äquivalent zum Suchbaum!

Bauminterface

Bauminterface

- *TreeNode* mit folgendem Interface:
  `initScript(Script), noOfChildren($), getChild(I $)`

Bauminterface

- *TreeNode* mit folgendem Interface:
  `initScript(Script), noOfChildren($), getChild(I $)`
- Verwaltung der Kinder

Bauminterface

- *TreeNode* mit folgendem Interface:
  `initScript(Script), noOfChildren($), getChild(I $)`
- Verwaltung der Kinder
  (Kein Kind doppelt erzeugen)

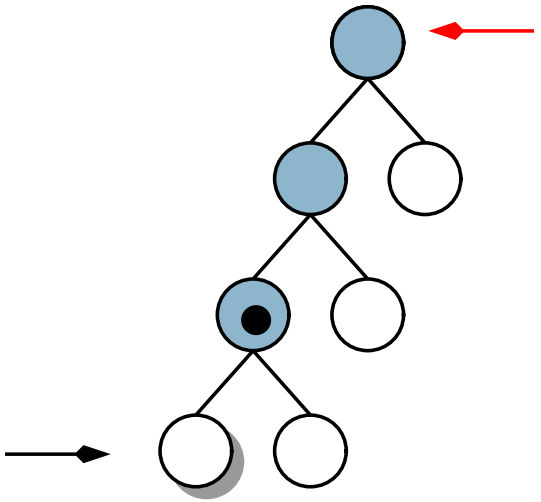
Bauminterface

- *TreeNode* mit folgendem Interface:
  `initScript(Script), noOfChildren($), getChild(I $)`
- Verwaltung der Kinder
  (Kein Kind doppelt erzeugen)
- Aufsammeln der Lösungen

Explorations-Strategie

```
class DFSClass
   from SearchClass
   meth DFSKids(M N Node)
      if M>N then skip
      else
         Kid = Node getChild(M $)
      in
         DFSClass, search(Kid)
         DFSClass, DFSKids(M+1 N Node)
      end
   end
   meth search(Node)
      case Node noOfChildren($) of
         0 then skip
      [] N then
         if Node getOpenChildren($) > 0 then
            DFSClass, DFSKids(1 N Node)
         end
      end
   end
end
```

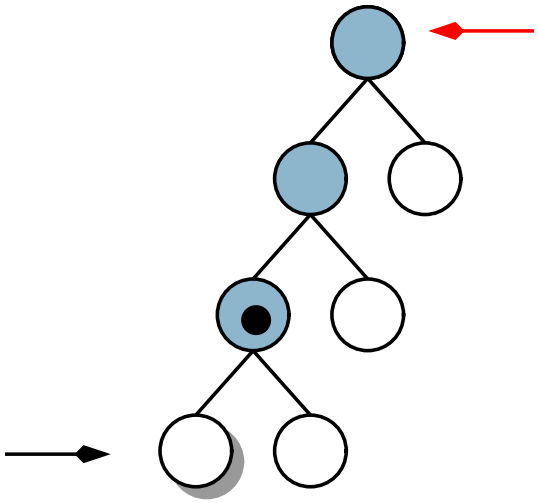
Beispiel



SpaceNode TreeNode

Beispiel

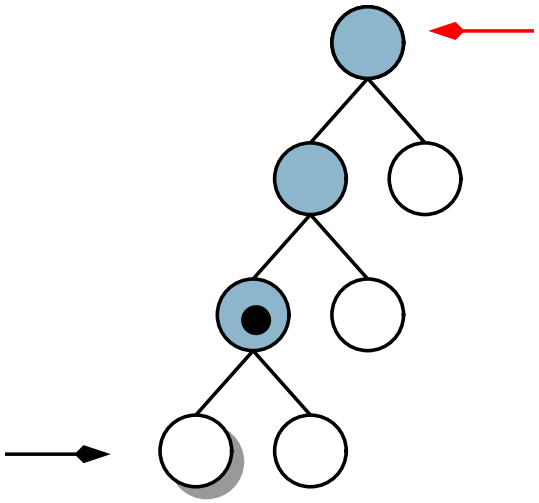
`noOfChildren`



SpaceNode TreeNode

Beispiel

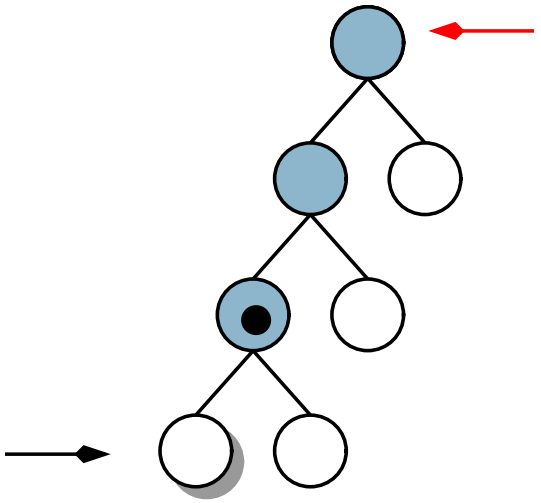
```
noOfChildren
   ask
```



SpaceNode TreeNode

Beispiel

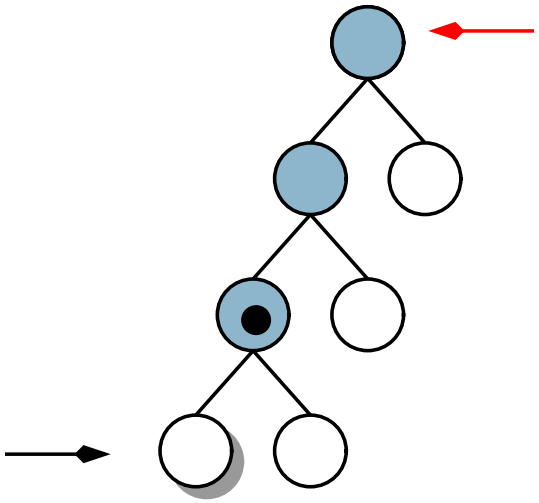
```
noOfChildren
  ask
    getSpace
```



SpaceNode TreeNode

Beispiel

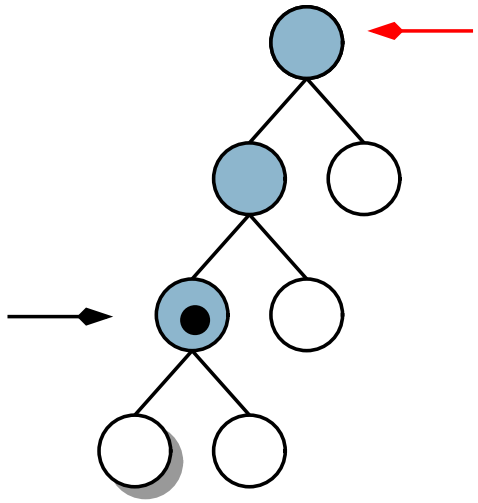
```
noOfChildren
  ask
    getSpace
      acquireSpace
```



SpaceNode TreeNode

Beispiel

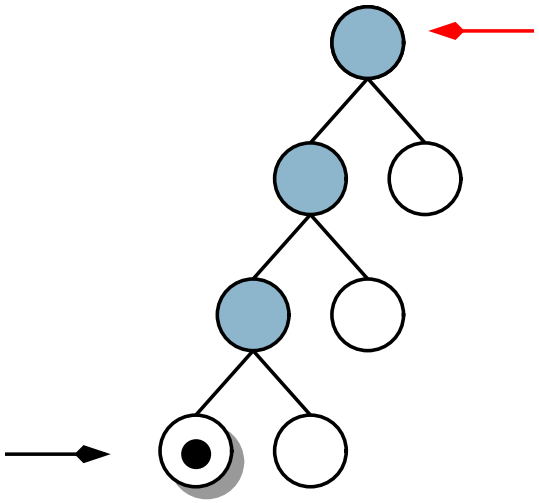
```
noOfChildren
  ask
    getSpace
      acquireSpace
        donateSpace
```



SpaceNode TreeNode

Beispiel

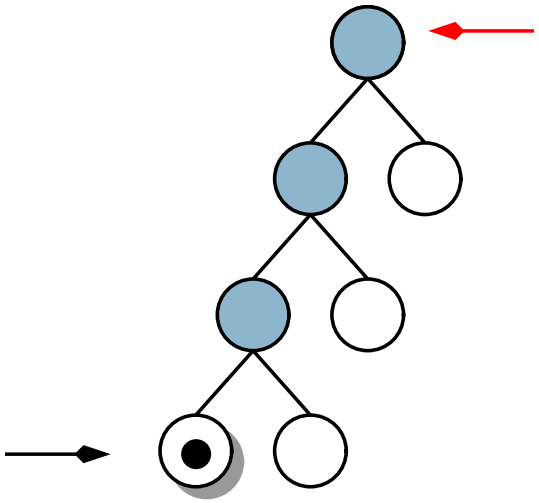
```
noOfChildren
  ask
    getSpace
      acquireSpace
```



SpaceNode TreeNode

Beispiel

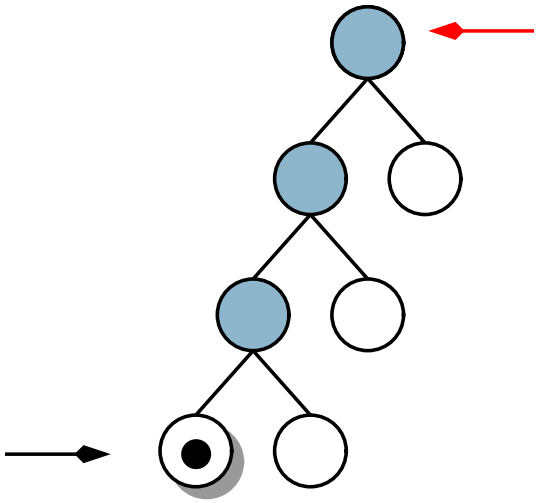
```
noOfChildren
  ask
    getSpace
      acquireSpace
        doCommits
```



SpaceNode TreeNode

Beispiel

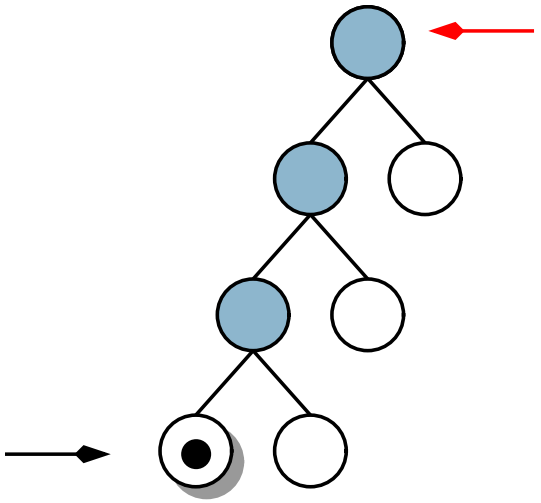
```
noOfChildren
  ask
    getSpace
      acquireSpace
```



SpaceNode TreeNode

Beispiel

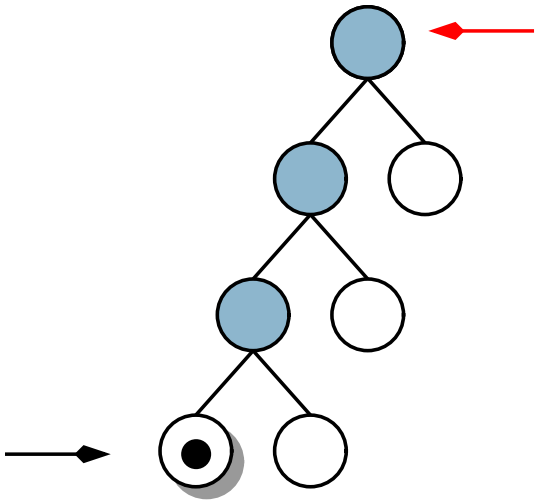
```
noOfChildren
  ask
    getSpace
```



SpaceNode TreeNode

Beispiel

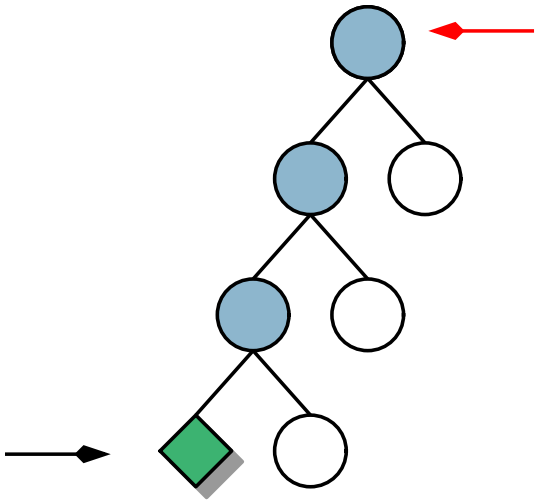
```
noOfChildren
   ask
```



SpaceNode TreeNode

Beispiel

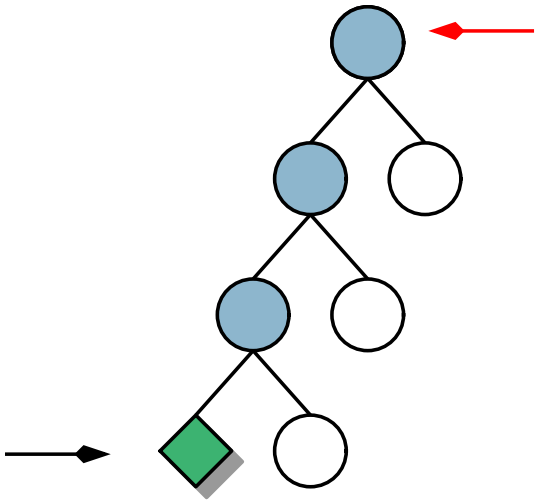
```
noOfChildren
   ask
```



SpaceNode TreeNode

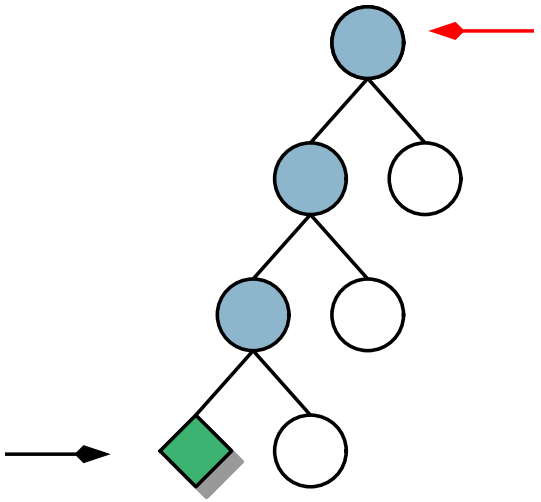
Beispiel

`noOfChildren`



SpaceNode TreeNode

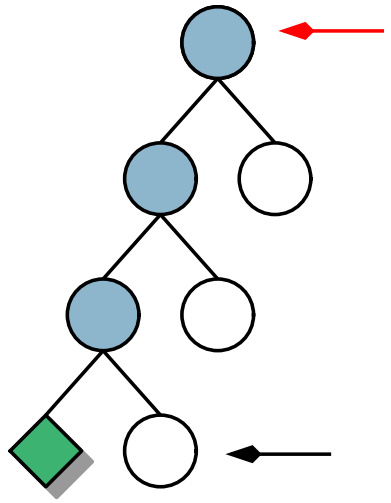
Beispiel



SpaceNode TreeNode

Beispiel

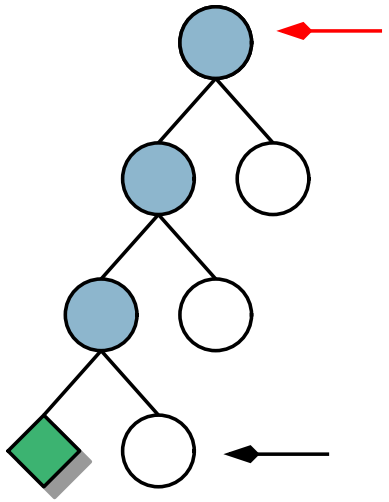
`noOfChildren`



SpaceNode TreeNode

Beispiel

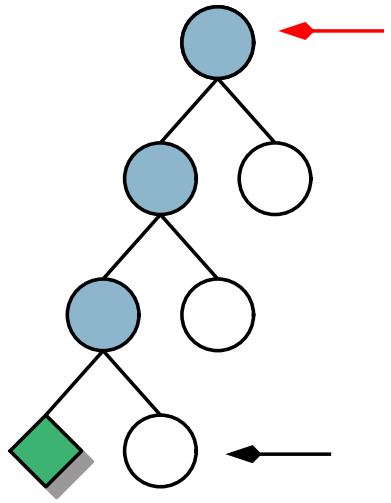
```
noOfChildren
   ask
```



SpaceNode TreeNode

Beispiel

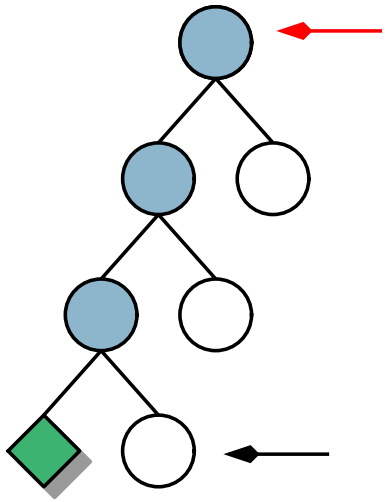
```
noOfChildren
  ask
    getSpace
```



SpaceNode TreeNode

Beispiel

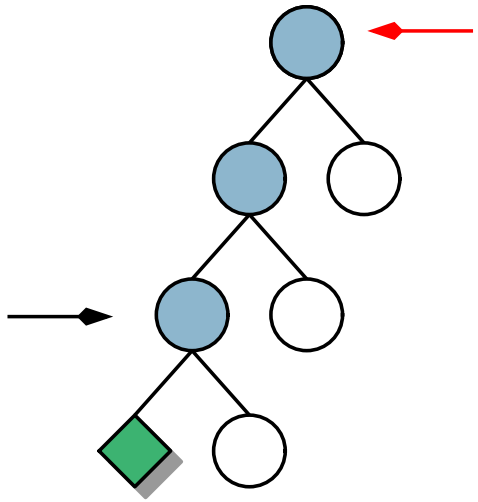
```
noOfChildren
  ask
    getSpace
      acquireSpace
```



SpaceNode TreeNode

Beispiel

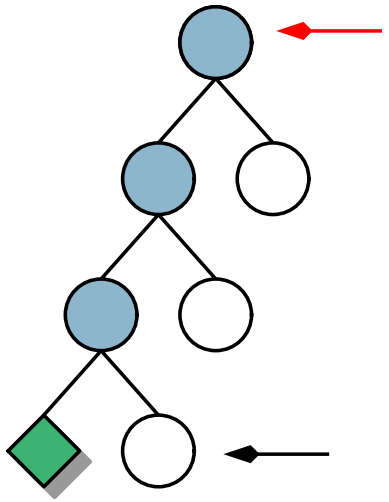
```
noOfChildren
  ask
    getSpace
      acquireSpace
        donateSpace
```



SpaceNode TreeNode

Beispiel

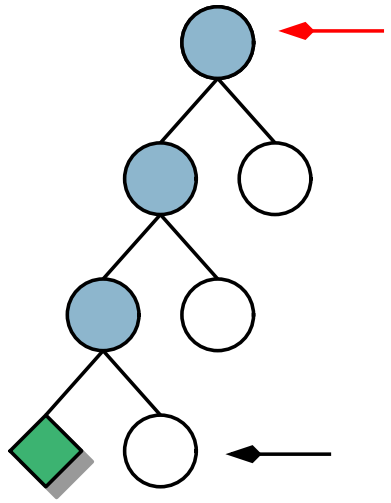
```
noOfChildren
  ask
    getSpace
      acquireSpace
```



SpaceNode TreeNode

Beispiel

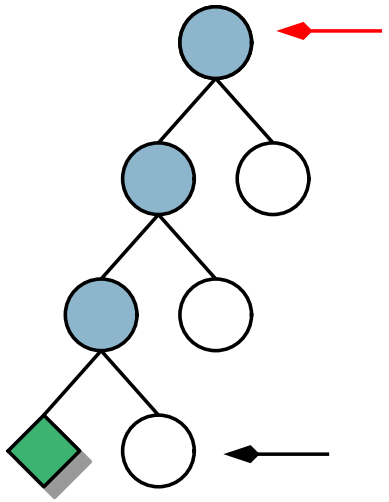
```
noOfChildren
  ask
    getSpace
```



SpaceNode TreeNode

Beispiel

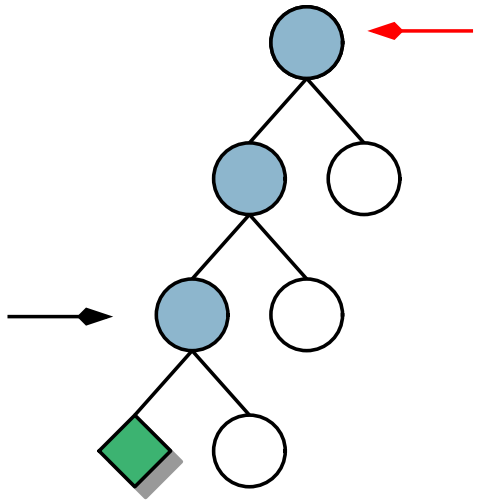
```
noOfChildren
  ask
    getSpace
      recompute
```



SpaceNode TreeNode

Beispiel

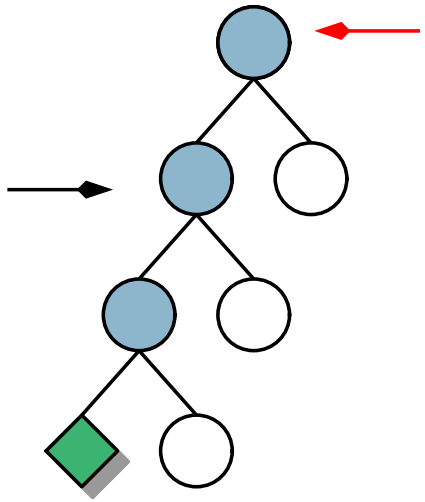
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
```



SpaceNode TreeNode

Beispiel

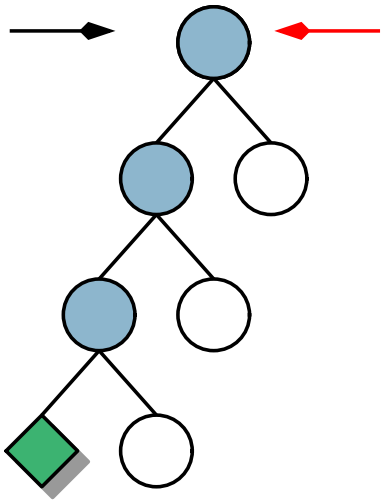
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          recompute
```



SpaceNode TreeNode

Beispiel

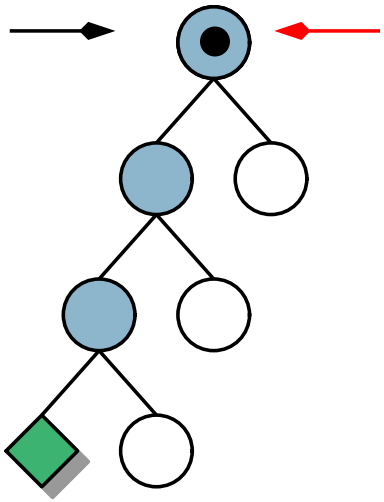
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          recompute
            recompute
```



SpaceNode TreeNode

Beispiel

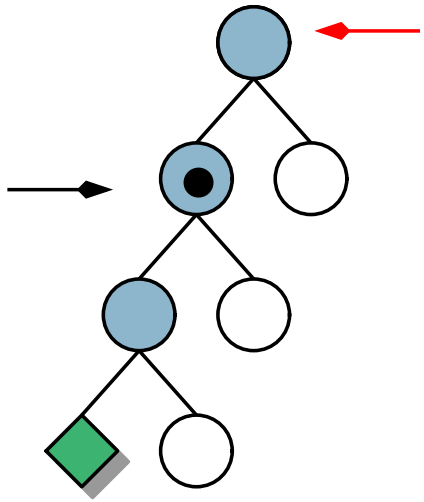
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          recompute
            recompute
```



SpaceNode TreeNode

Beispiel

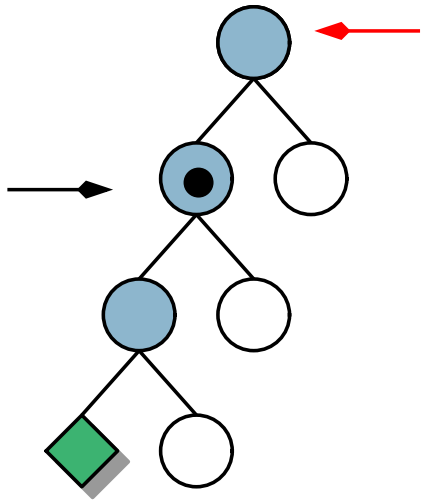
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          recompute
```



SpaceNode TreeNode

Beispiel

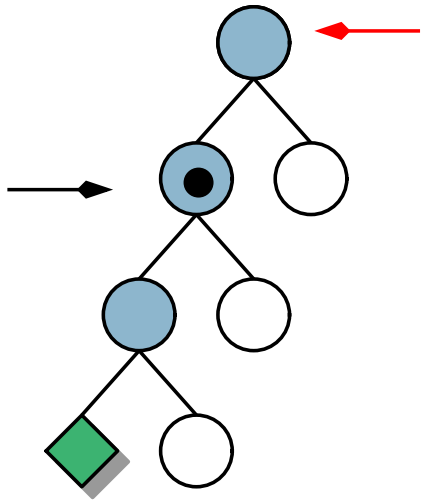
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          recompute
            doCommits
```



SpaceNode TreeNode

Beispiel

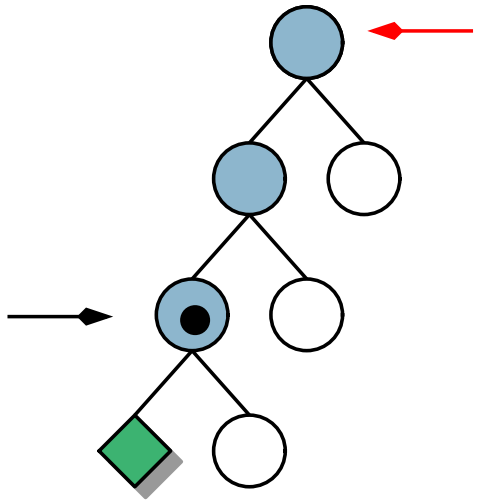
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          recompute
```



SpaceNode TreeNode

Beispiel

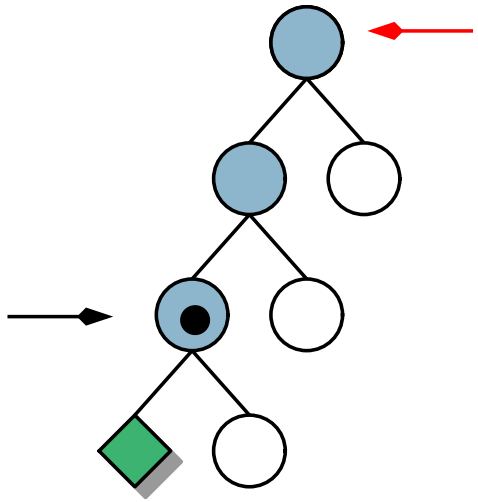
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
```



SpaceNode TreeNode

Beispiel

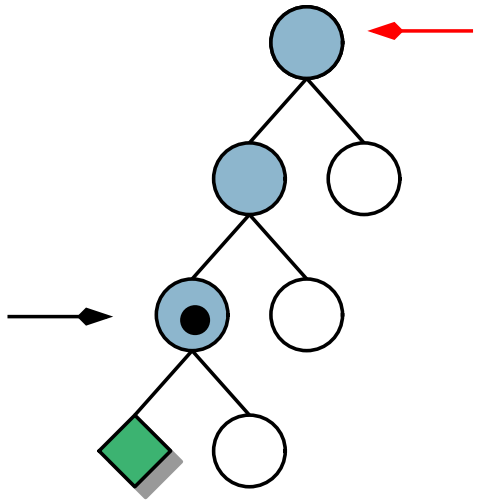
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
          doCommits
```



SpaceNode TreeNode

Beispiel

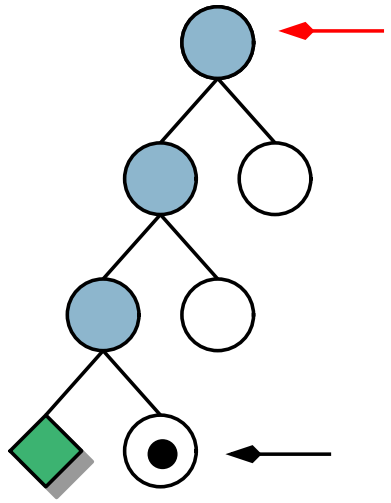
```
noOfChildren
  ask
    getSpace
      recompute
        recompute
```



SpaceNode TreeNode

Beispiel

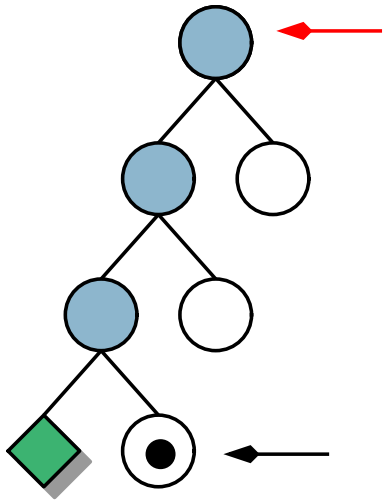
```
noOfChildren
  ask
    getSpace
      recompute
```



SpaceNode TreeNode

Beispiel

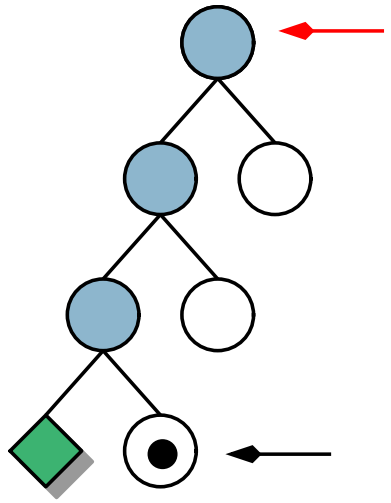
```
noOfChildren
  ask
    getSpace
      recompute
        doCommits
```



SpaceNode TreeNode

Beispiel

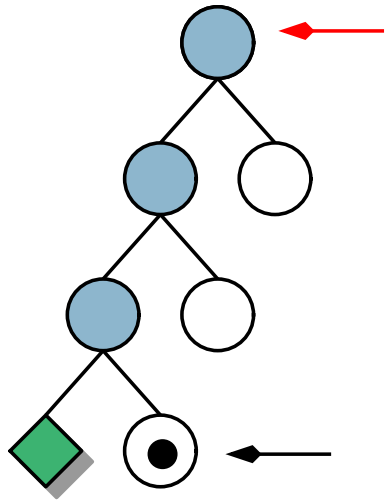
```
noOfChildren
  ask
    getSpace
      recompute
```



SpaceNode TreeNode

Beispiel

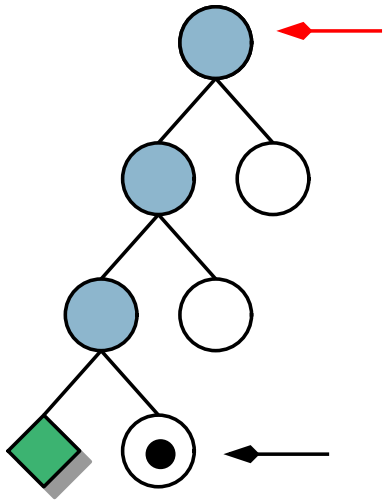
```
noOfChildren
  ask
    getSpace
```



SpaceNode TreeNode

Beispiel

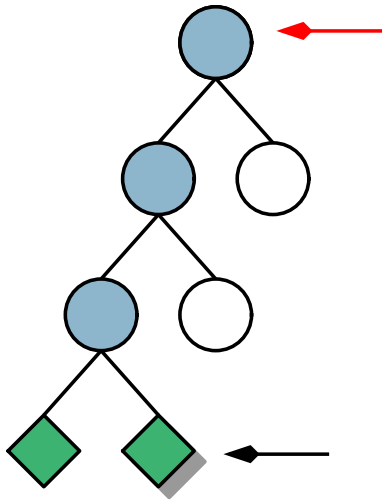
```
noOfChildren
   ask
```



SpaceNode TreeNode

Beispiel

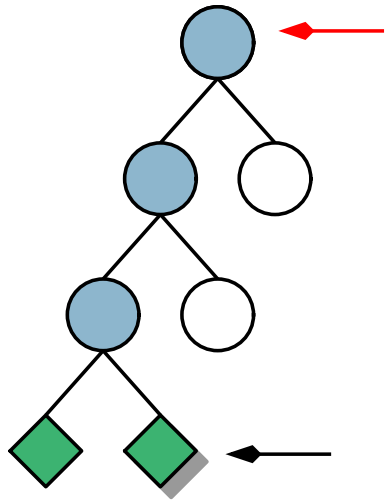
```
noOfChildren
   ask
```



SpaceNode TreeNode

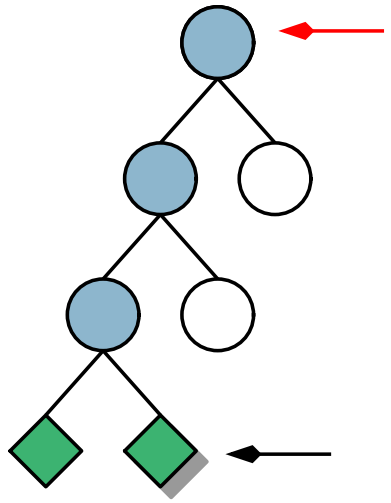
Beispiel

`noOfChildren`



SpaceNode TreeNode

Beispiel



SpaceNode TreeNode

Schwierigkeiten

Schwierigkeiten

- *Branch & Bound*: Buchhaltung über letzte Lösung

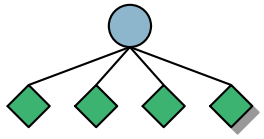
Schwierigkeiten

- *Branch & Bound*: Buchhaltung über letzte Lösung
- Suspension: Nebenläufige Auflösung

Schwierigkeiten

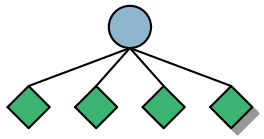
- *Branch & Bound*: Buchhaltung über letzte Lösung
- Suspension: Nebenläufige Auflösung
- *Last Alternative Optimization* unabhängig von Exploration

4 Evaluation



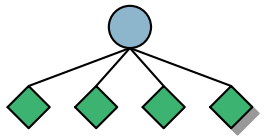
4 Evaluation

- Als orthogonal identifizierte Teile lassen sich auch orthogonal implementieren



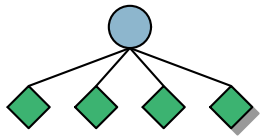
4 Evaluation

- Als orthogonal identifizierte Teile lassen sich auch orthogonal implementieren
- Implementierte Suchstrategien:



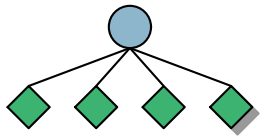
4 Evaluation

- Als orthogonal identifizierte Teile lassen sich auch orthogonal implementieren
- Implementierte Suchstrategien:
  DFS, ID, LDS



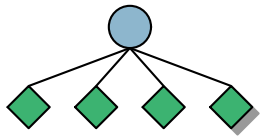
4 Evaluation

- Als orthogonal identifizierte Teile lassen sich auch orthogonal implementieren
- Implementierte Suchstrategien:
  DFS, ID, LDS
  - Interface mächtig genug



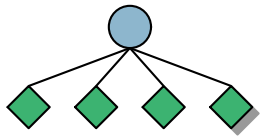
4 Evaluation

- Als orthogonal identifizierte Teile lassen sich auch orthogonal implementieren
- Implementierte Suchstrategien:
  DFS, ID, LDS
  - ▷ Interface mächtig genug
- Zur Zeit noch ca. Faktor 1.6-5 langsamer als Bibliotheks-Routinen



4 Evaluation

- Als orthogonal identifizierte Teile lassen sich auch orthogonal implementieren
- Implementierte Suchstrategien:
  DFS, ID, LDS
  - ▷ Interface mächtig genug
- Zur Zeit noch ca. Faktor 1.6-5 langsamer als Bibliotheks-Routinen (u.a. durch höheren Speicherverbrauch)



Zusammenfassung

Zusammenfassung

- Neue Suchstrategien leicht zu implementieren

Zusammenfassung

- Neue Suchstrategien leicht zu implementieren
- Transparente, automatische Recomputation

Zusammenfassung

- Neue Suchstrategien leicht zu implementieren
- Transparente, automatische Recomputation
- Korrekte Implementierung von *Branch & Bound*

Zusammenfassung

- Neue Suchstrategien leicht zu implementieren
- Transparente, automatische Recomputation
- Korrekte Implementierung von *Branch & Bound*
  (Unabhängig von Recomputation und Explorations-Strategie!)

Zusammenfassung

- Neue Suchstrategien leicht zu implementieren
- Transparente, automatische Recomputation
- Korrekte Implementierung von *Branch & Bound*
  (Unabhängig von Recomputation und Explorations-Strategie!)
- Modulares Design

Zusammenfassung

- Neue Suchstrategien leicht zu implementieren
- Transparente, automatische Recomputation
- Korrekte Implementierung von *Branch & Bound*
  (Unabhängig von Recomputation und Explorations-Strategie!)
- Modulares Design
  (Portierung nach Gtk, Wiederverwendung der Grafikroutinen)

Ausblick

Ausblick

- Mehr Explorations-Strategien implementieren

Ausblick

- Mehr Explorations-Strategien implementieren
- Verteilte Suche

Ausblick

- Mehr Explorations-Strategien implementieren
- Verteilte Suche
  (Nicht klar, wie die in das Framework passt)

Ausblick

- Mehr Explorations-Strategien implementieren
- Verteilte Suche
  (Nicht klar, wie die in das Framework passt)
- Gtk-Oberfläche

Ausblick

- Mehr Explorations-Strategien implementieren
- Verteilte Suche
  (Nicht klar, wie die in das Framework passt)
- Gtk-Oberfläche
  (teilweise fertig)

Ausblick

- Mehr Explorations-Strategien implementieren
- Verteilte Suche
  (Nicht klar, wie die in das Framework passt)
- Gtk-Oberfläche
  (teilweise fertig)
  - **Ersetzen des Explorers**

ENDE