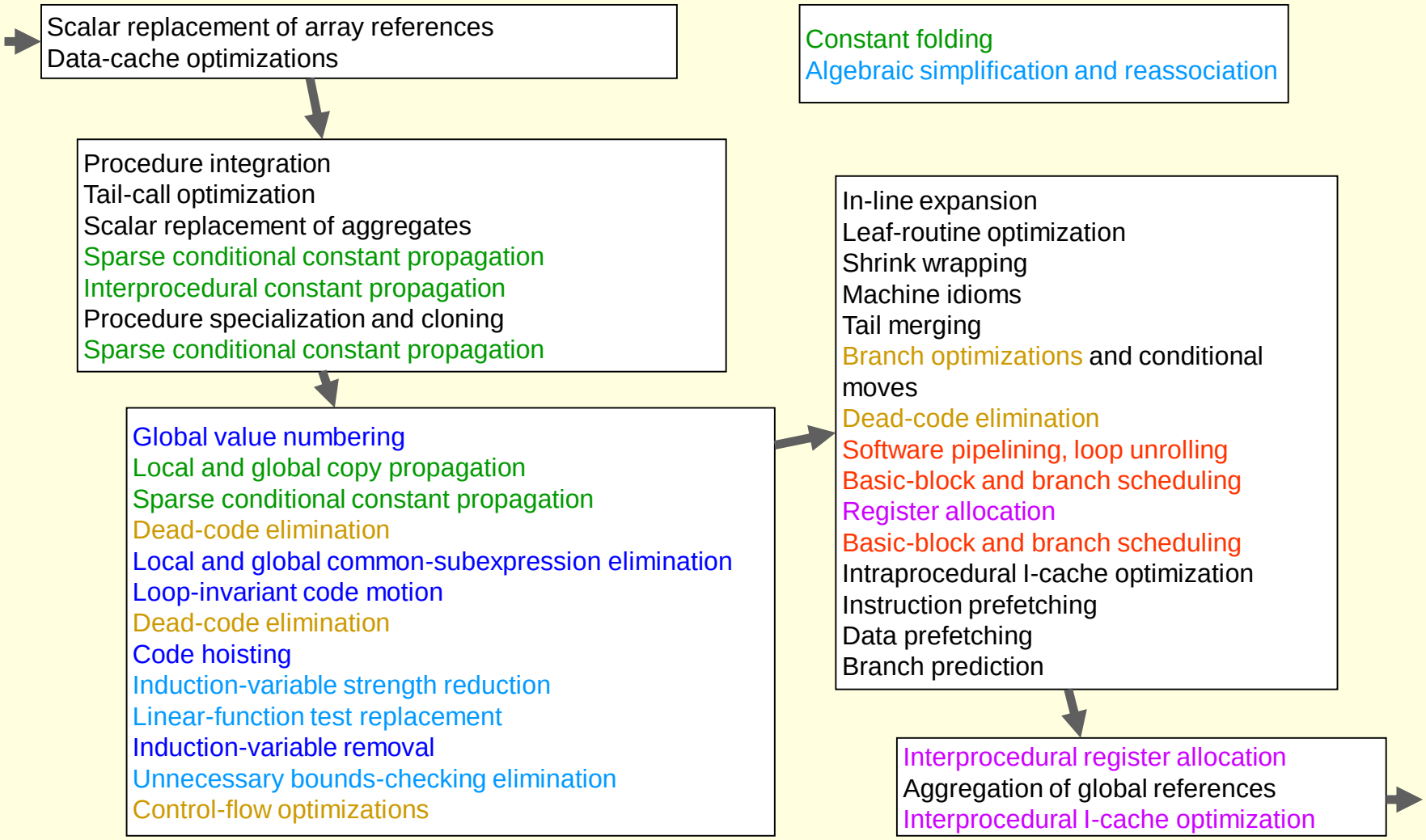
Další optimalizace

Architektura překladače

- Detailní pohled akademika (pouze optimalizace)
  - Muchnick: Advanced Compiler Design and Implementation



Další optimalizace

- **Částečné vyhodnocení**
  - Část požadovaného výpočtu je vyhodnocována již překladačem
  - Výpočet konstantních výrazů
    - Constant-expression evaluation (constant folding)
  - Výpočet podmíněně konstantních výrazů
    - Sparse conditional constant propagation
- **Algebraické úpravy**
  - Využití algebraických identit ke zjednodušení kódu
  - Algebraické úpravy výrazů
  - Redukce síly v cyklech
    - Strength reduction
  - Odstranění zbytečných kontrol mezí

Další optimalizace

- **Odstranění redundance**
  - Nahrazení opakovaných výpočtů uložením výsledku
  - Copy propagation
  - Lokální/globální eliminace společných podvýrazů
    - Common-subexpression elimination
  - Přesun invariantního kódu z cyklu
    - Loop-invariant code motion
  - Partial-redundancy elimination
    - Lazy code motion

- **Odstranění neužitečného kódu**
  - Odstranění mrtvého kódu
    - Dead-code elimination
  - Odstranění nedosažitelného kódu
    - Unreachable-code elimination
  - Optimalizace skoků
    - Jump optimization

Částečné vyhodnocení

- Výpočet (pod)výrazů obsahujících pouze konstanty
  - Constant-expression evaluation
  - Obvykle prováděn již front-endem
- Určení proměnných s konstantním obsahem
  - Constant folding
- Určení proměnných s podmíněně konstantním obsahem
  - Sparse conditional constant propagation
  - Upravuje control-flow!

```
a = b + (4 * 10);
```

```
c = 4 * 10;
d = c + 5;
e = f + (d * 2);
```

```
if ( g > h )
  i = 1;
else
  i = 0;
j = i + 1;
if ( j > 1 )
  k = k + 1;
```

Částečné vyhodnocení

- Výpočet (pod)výrazů obsahujících pouze konstanty
  - Constant-expression evaluation
  - Obvykle prováděn již front-endem
- Určení proměnných s konstantním obsahem
  - Constant folding
- Určení proměnných s podmíněně konstantním obsahem
  - Sparse conditional constant propagation
  - Upravuje control-flow!

- Integrace procedur generuje nové příležitosti pro částečné vyhodnocení

```
void f( int i, bool f)
{
  int j = i + 1;
  if ( f )
    g( j);
  else
    h( j);
}

f( k + 1, false);
```

Částečné vyhodnocení


```
a = b + 40;



c = 40;
d = 45;
e = f + 90;


if ( g > h )
{
  i = 1;
  j = 2;
  k = k + 1;
}
else
{
  i = 0;
  j = 1;
}
```

```
a = b + (4 * 10);



c = 4 * 10;
d = c + 5;
e = f + (d * 2);


if ( g > h )
  i = 1;
else
  i = 0;
j = i + 1;
if ( j > 1 )
  k = k + 1;
```

Algebraické úpravy

- Algebraické úpravy výrazů
  - Většinou v souvislosti s přítomností konstant
  - Důležité pro ukazatelovou aritmetiku (přístup k polím)
  - Úprava do kanonického tvaru
- Redukce síly v cyklech
  - Důležité pro přístup k polím
- Odstranění zbytečných kontrol mezí
- Machine idioms
  - Instrukce provádějící speciální kombinace nebo varianty operací

```
a = ((b * 3) + 7) * 5

a = b * 15 + 35
```

```
for ( i = 1; i < 10; ++i )
  a[ 4 * i] = 0;

for ( j = 4; j < 40; j += 40 )
  a[ j] = 0;
```

```
int b[ 10];
for ( i = 0; i < 10; ++i )
  b[ i] = 0;
```

Algebraické úpravy

- Převedení control-flow na algebraické operace
  - Conditional move
    - Ušetří podmíněné skoky
    - Může přidat zbytečné operace
  - Užitečnost obtížně odhadnutelná
    - Cena podmíněných skoků závisí na úspěšnosti predikce
    - Překladač úspěšnost nedokáže odhadnout
    - Profilem řízené optimalizace

```
if ( a > b )
  c = d + e;
```

```
CMP t,a,b
ADD u,d,e
CMOV t,c,u
```

Odstranění redundance

- Copy propagation
- Lokální/globální eliminace společných podvýrazů
- Přesun invariantního kódu z cyklu
  - Častý výskyt u přístupu k polím
- Partial-redundancy elimination
  - Lazy code motion
- Integrace procedur generuje nové redundance

```
b = a;
c = b;          // c = a;

c = a + b;
d = a + b;      // d = c;

for ( i = 0; i < 10; ++i)
  a[ i] = k + l;


if ( a < b )
  c = d + e;
f = d + e;
```

Odstranění neužitečného kódu

- Odstranění mrtvého kódu
  - Kód, jehož efekt nebude využit
    - Přiřazení do proměnných, které již nebudou čteny
- Odstranění nedosažitelného kódu
  - Kód, ke kterému nevede cesta
- Optimalizace skoků
  - Skoky na skoky apod.
- Řeší především chyby vyprodukované předchozími fázemi
  - Aplikuje se opakovaně

```
a = b + 40;



c = 40;
d = 45;
e = f + 90;


if ( g > h )
{
  i = 1;
  j = 2;
  k = k + 1;
}
else
{
  i = 0;
  j = 1;
}
```

Přesouvání kódu

- Code hoisting
  - Provedení operace dříve, než předepisoval původní program
  - Omezeno závislostmi
  - „very busy“ expression = operace, která bude provedena v každém pokračování
  - Algoritmus: Lazy code motion

- Užitečnost úpravy je nejistá
  - Paralelismus
  - Omezený počet registrů

```
if ( g > h )
{
  for ( i = 1; i < k; ++ i)
  { a[ i] = x + y;
  }
  u = x + y;
}
else
{
  z = x + y;
}
```

Optimalizace režie volání

- Integrace procedur
  - a.k.a inline-expansion
- Listové procedury
  - Nevolají žádné další
  - Není třeba kompletní prolog a epilog procedury
  - Užitečné zejména v přítomnosti výjimek
- Shrink wrapping
  - Přesouvání prologu a epilogu
  - Ve větvích, které nevolají další procedury, se prolog a epilog anihilují

- Tail merging
  - Ztotožnění identických konců procedur
    - Poslední BB často obsahuje pouze epilog a je tudíž shodný
  - Šetří velikost kódu
    - Zlepšuje využití I-cache

Interprocedurální úpravy

- Specializace procedur
  - Procedury se naklonují
  - Jednotlivé klony se přizpůsobují okolnostem, které panují při jejich volání
    - Speciální tvary argumentů
    - Konstantní argumenty
    - Aliasing
- Interprocedurální alokace registrů
  - Volací konvence se upravuje podle místních podmínek

- Efekt je obdobný jako u integrace procedur
  - Specializace vede k menší expanzi kódu
  - Specializace se obtížněji řídí
  - Integrace odstraňuje režii volání
  - Integrace umožňuje další optimalizace

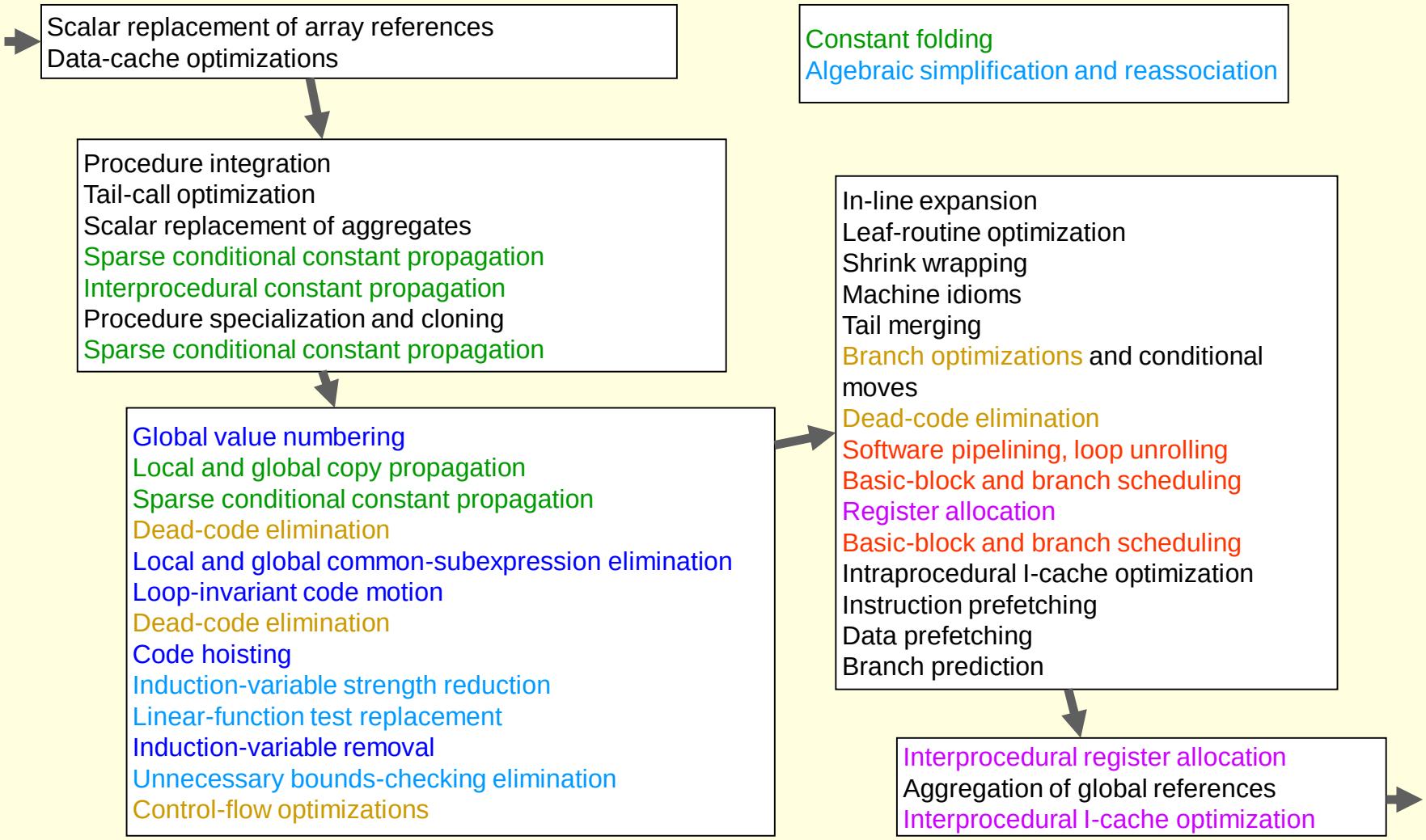
Nápověda pro procesor

- Intraprocedural I-cache optimization
  - Využití atomicity cache-line
  - Serializace BB tak, aby chování CPU vedlo k minimálnímu počtu výpadků I-cache
- Instruction prefetching
- Data prefetching
  - Využití speciálních instrukcí pro nedestruktivní čtení

- Branch prediction
  - Generování nápovědy pro branch prediction

Architektura překladače

- Detailní pohled akademika (pouze optimalizace)
  - Muchnick: Advanced Compiler Design and Implementation



Architektura překladače

- Realita
  - GNU Compiler Collection Internals