Vérification formelle d'un algorithme d'allocation de registres par coloration de graphes

Sandrine Blazy, Benoît Robillard, Éric Soutif

JFLA 2008 et 2014

UNIVERSITÉ DE RENNES 1 — UMR IRISA — Inria informatics mathematics

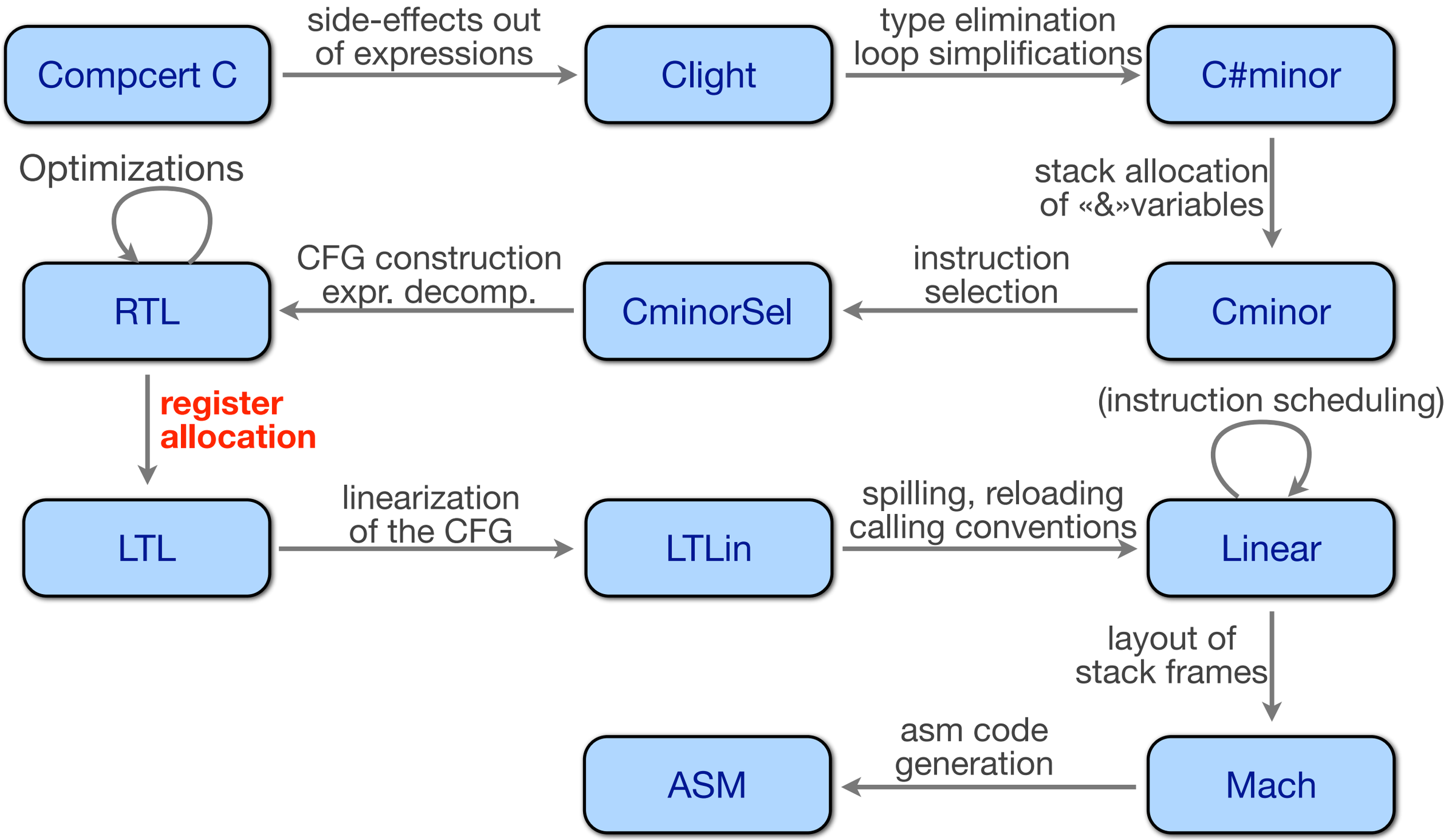
Le projet CompCert

But : développer et prouver correct un compilateur réaliste utilisable pour du logiciel embarqué critique

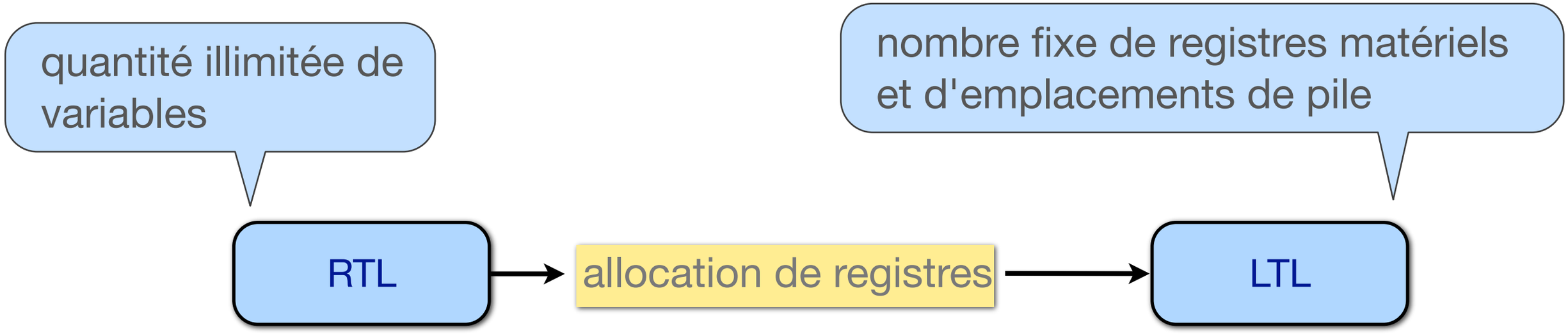
- (vaste sous-ensemble du) langage C
- code assembleur pour des processeurs répandus (PowerPC, ARM, x86)
- produisant un code raisonnablement efficace (→ quelques optimisations)

Utilisation de Coq pour automatiser la preuve de préservation sémantique et aussi pour implémenter la plupart du compilateur.
(Exécutable via l'extraction automatique vers Caml.)

Le compilateur formellement vérifié CompCert

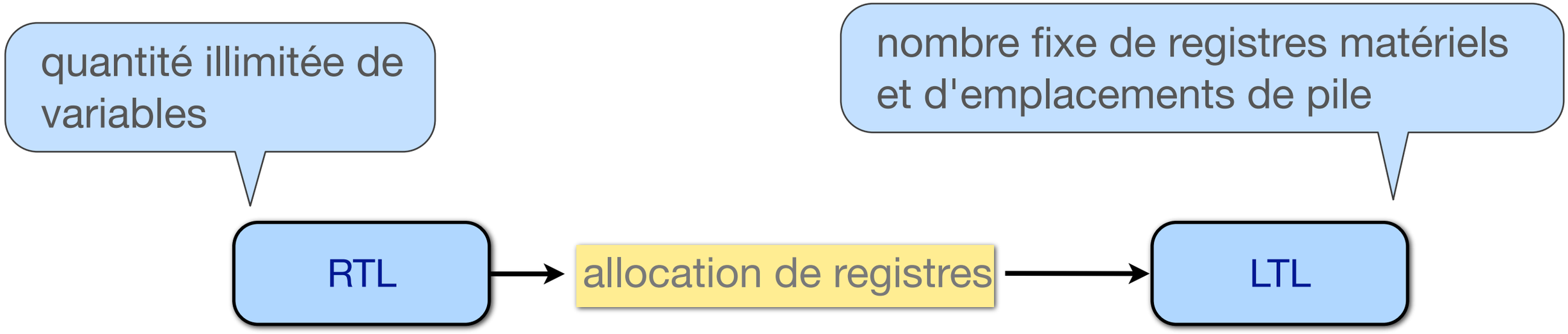


Allocation de registres



But : maximiser l’utilisation des registres

Allocation de registres



But : maximiser l’utilisation des registres

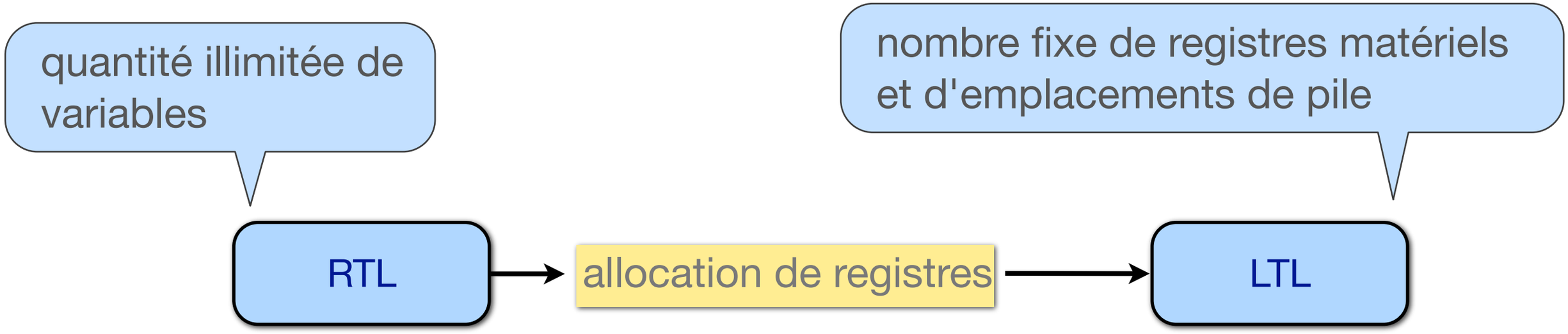
Approche naïve

- Affecter les N registres matériels aux N variables les plus utilisées; affecter les emplacements de pile aux autres variables.

Meilleure approche

- Remarquer qu'un même registre matériel peut être affecté à différentes variables, pourvu qu'elles ne soient jamais utilisées simultanément.

Allocation de registres



But : maximiser l’utilisation des registres

Deux tâches principales :

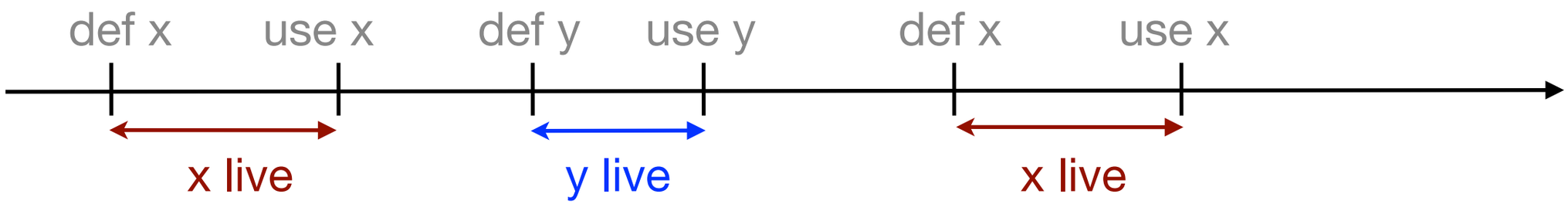
- vidage de registres en mémoire
- fusion de registres

Heuristiques combinant ces deux tâches

Analyse de vivacité pour l'allocation de registres

Deux variables x et y interfèrent si elles sont toutes deux vivantes en un point de programme.

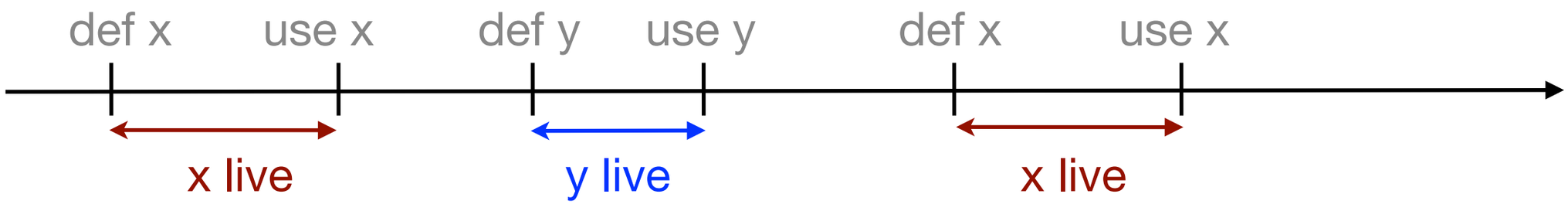
Si x et y n'interfèrent pas, elles peuvent partager le même registre ou emplacement de pile.



Analyse de vivacité pour l'allocation de registres

Deux variables x et y interfèrent si elles sont toutes deux vivantes en un point de programme.

Si x et y n'interfèrent pas, elles peuvent partager le même registre ou emplacement de pile.

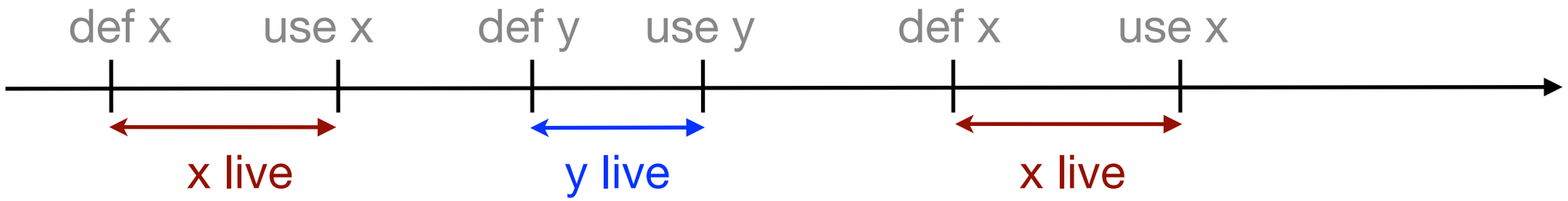


- Déterminer le nombre minimal de registres nécessaire pour colorer le graphe représentant la relation d'interférence.

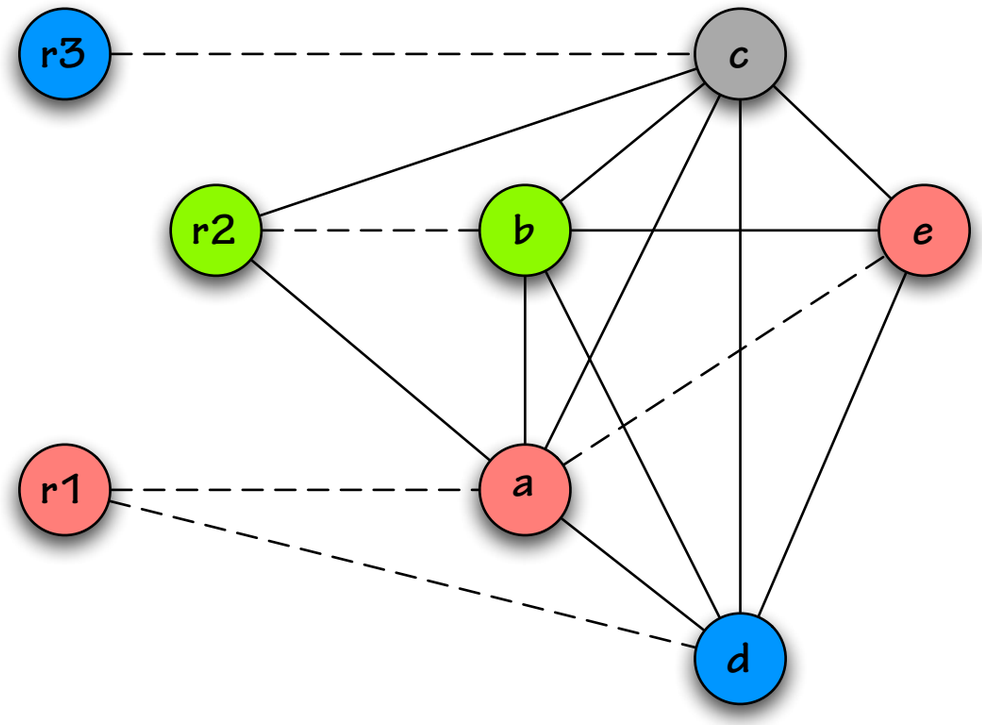
Analyse de vivacité pour l'allocation de registres

Deux variables x et y interfèrent si elles sont toutes deux vivantes en un point de programme.

Si x et y n'interfèrent pas, elles peuvent partager le même registre ou emplacement de pile.



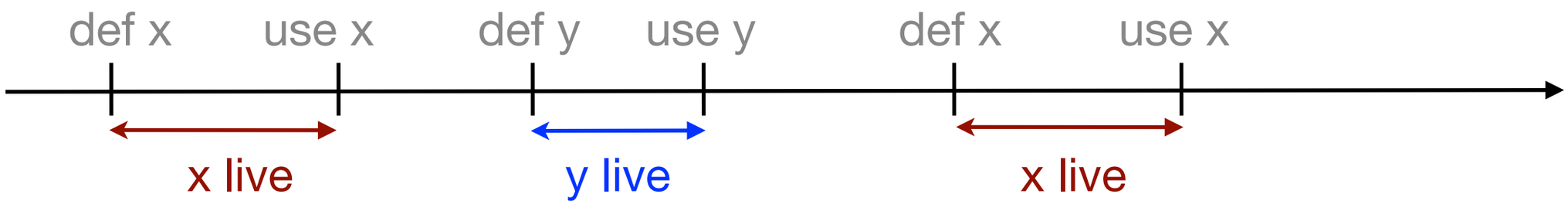
- Déterminer le nombre minimal de registres nécessaire pour colorer le graphe représentant la relation d'interférence.



Analyse de vivacité pour l'allocation de registres

Deux variables x et y interfèrent si elles sont toutes deux vivantes en un point de programme.

Si x et y n'interfèrent pas, elles peuvent partager le même registre ou emplacement de pile.

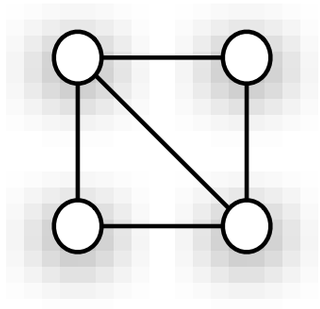


- Déterminer le nombre minimal de registres nécessaire pour colorer le graphe représentant la relation d'interférence.
- Si ce nombre est ≤ nombre de registres matériels, nous obtenons une allocation de registres parfaite.
- Sinon, le coloriage est un bon point de départ pour déterminer quelles variables vont dans quels registres.

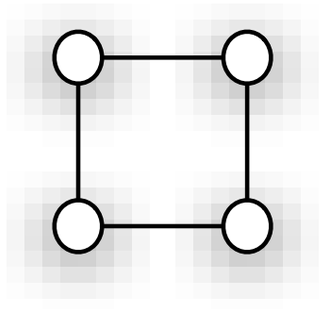
L’article JFLA 2008

Formalisation d’un algorithme de coloration gourmande, intégré à CompCert

- Teste la colorabilité d’un graphe triangulé



chordal



non chordal

- Un début de bibliothèque de graphes
  - clique
  - sommet simplicial (dont les voisins forment une clique)
  - ordre d’élimination simplicial (permutation des sommets t.q. chaque sommet est simplicial)

Un graphe est triangulé ssi il possède une ordre d’élimination simplicial.

De plus, tout sommet simplicial peut être le premier sommet d’un ordre élimination simplicial. Cet ordre est utilisé pour colorer le graphe.

L’article JFLA 2008

Preuves :

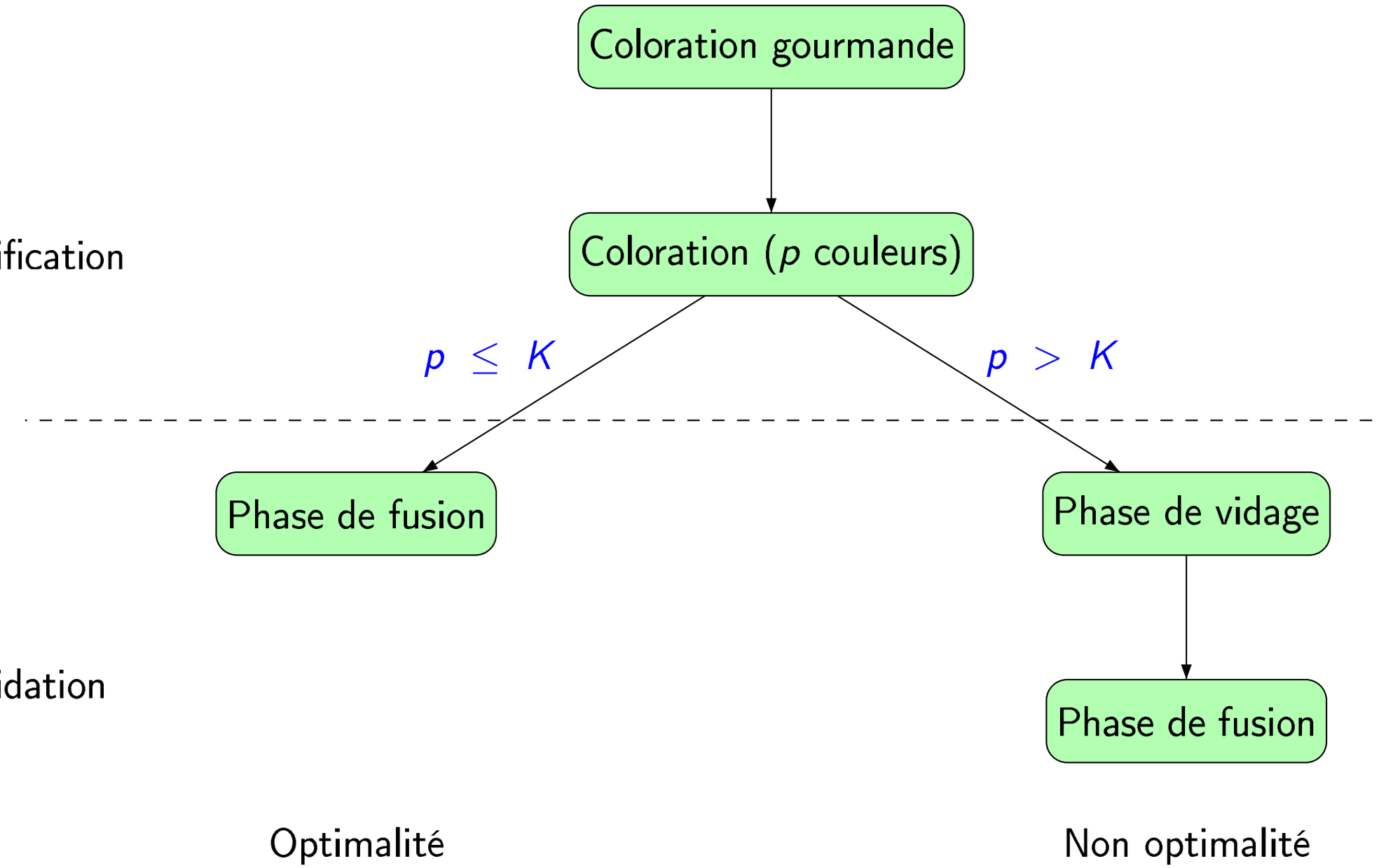
- correction de la coloration gourmande
- optimalité de la coloration gourmande (pour des graphes triangulés)
  La plus grande couleur utilisée par toute coloration est supérieure à la plus grande couleur renvoyée par la coloration de l'algorithme.

Objectifs :

- utiliser des algorithmes efficaces connus en OC
  (ex. vidage et fusion par PLNE)
- écrire et vérifier en Coq certaines parties

L'article JFLA 2008

Certification



Validation

Optimalité

Non optimalité

Allocation de registres : algorithmes

1. Analyse de vivacité
   Résolution d'équations data-flow arrière
   (algorithme de Kildall)

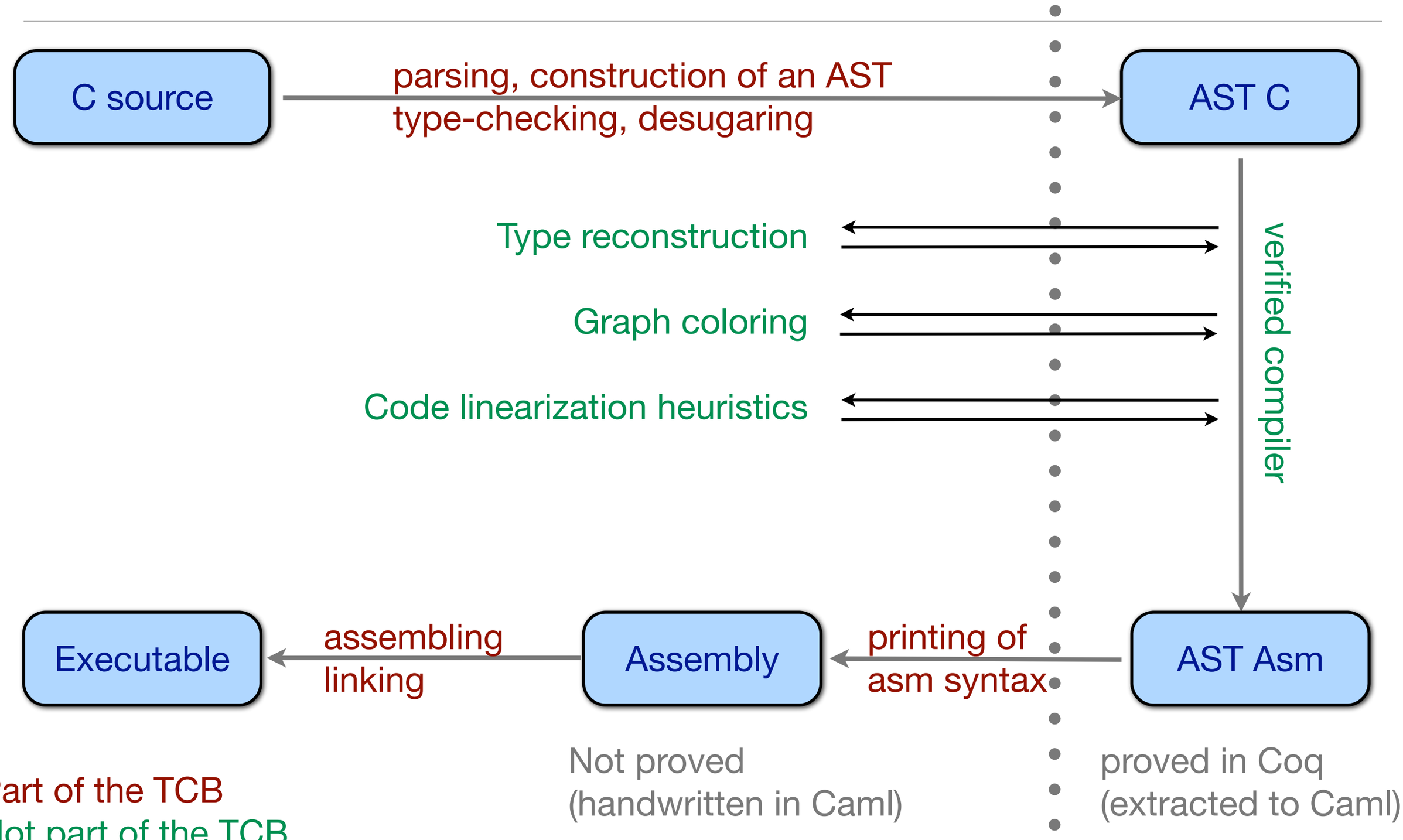
2. Construction du graphe d'interférence
   Ajout d'arcs pour chaque instruction

Construire une fonction
color: Variables → Register + Stackslot
telle que color(x) ≠ color(y) si x et y interfèrent

3. Coloriage du graphe interférence
   - Coloration gourmande + PLNE (graphes triangulés)
   - Heuristique de coloriage d'Appel & George (IRC)

4. Réécriture du code
   Remplacer toutes les variables par leur couleur

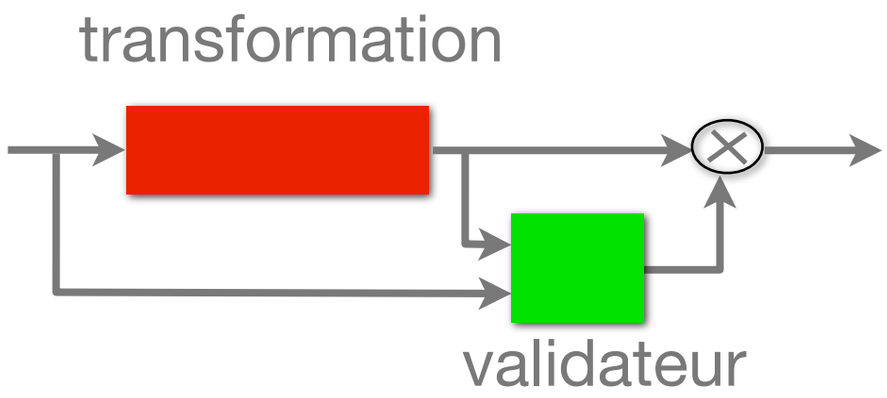
Le compilateur CompCert



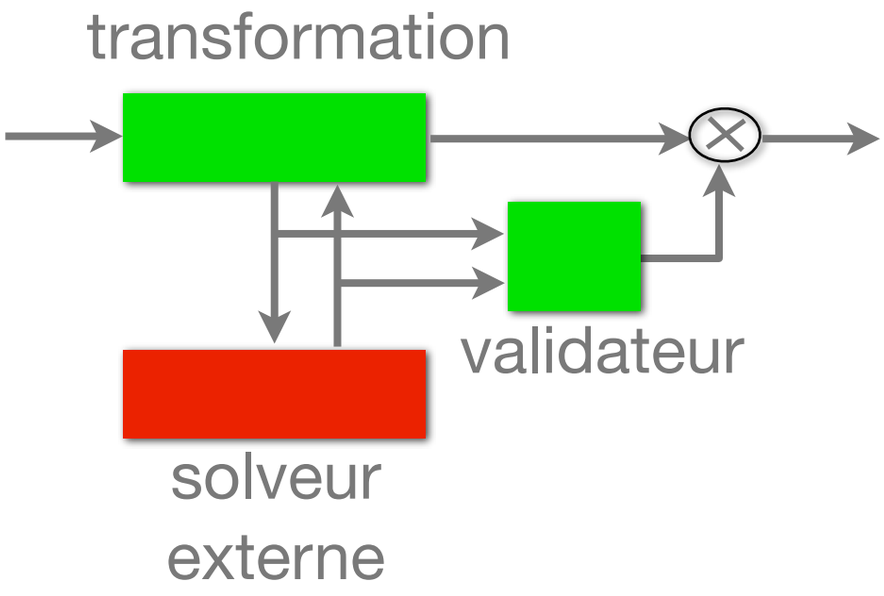
Schémas de vérification
(pour chaque passe de compilation)

Transformation vérifiée

Validateur vérifié



Solveur externe avec transformation vérifiée



= formellement vérifié

= non vérifié

Validateurs et vérificateurs vérifiés

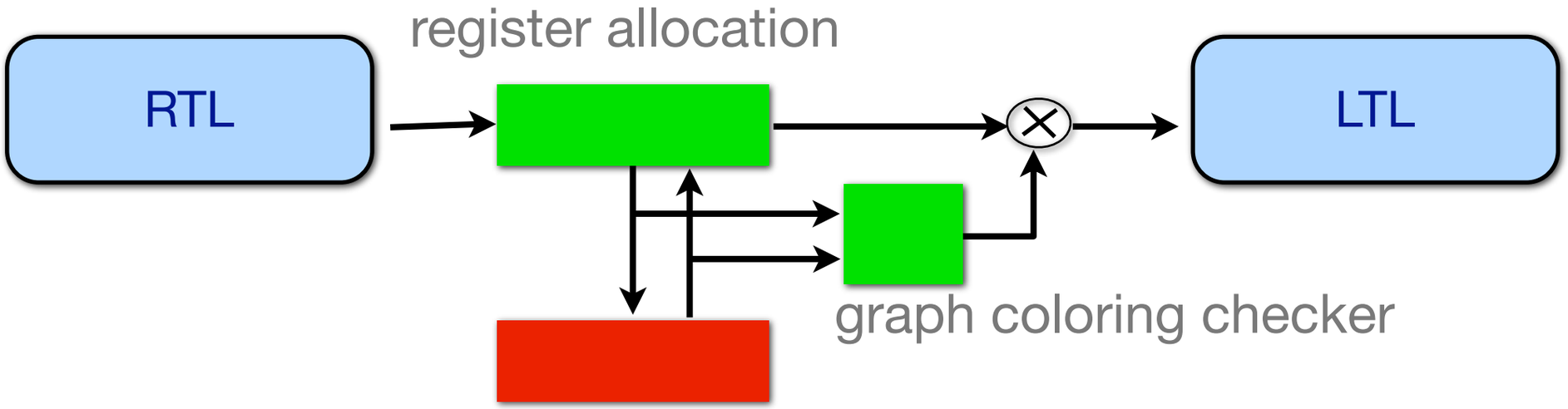
Validateur ou vérificateur vérifié

- Moins à prouver (si le validateur est plus simple que la transformation).
- Garanties fortes de correction, mais pas de complétude en général.
- Validateur réutilisable pour plusieurs variantes d'une transformation.
- Peut être efficace (si suffisamment peu coûteux pour être utilisé à chaque compilation).

Exemples de validateurs vérifiés

- Allocation de registres
- Lazy code motion

Prouver l'algorithme IRC ou pas ?



Il faut montrer que $(x,y) \in G \Leftrightarrow color(x) \neq color(y)$

- Le validateur énumère tous les arcs (x,y) de G et échoue si color(x) = color(y).
- La preuve de correction du vérificateur est triviale.

La suite de la formalisation

«Formal verification of coalescing graph-coloring register allocation»
Sandrine Blazy, Benoît Robillard, Andrew Appel
ESOP 2010

- Une implémentation de référence d'un algorithme très répandu.
  (Des erreurs ont été signalées dans plusieurs implémentations.)
- Preuve de correction + preuve de terminaison
- Formalisation de structures de données avancées + raisonnements intensifs sur les graphes.
- Plus de 6000 lignes de Coq

«Vérification formelle et optimisation de l'allocation de registres»
Benoît Robillard.
Thèse soutenue au CNAM en novembre 2010

Différentes allocations de registres pour CompCert

- 4300 lignes IRC validé, le reste vérifié
  Première passe validée de CompCert.

- +6000 lignes IRC vérifié
  [Blazy, Robillard, Appel, ESOP 2010]

- 900 lignes Allocation de registres validée, avec vidage de registres et live-range splitting améliorés [Rideau, Leroy, CC 2010]
  - La preuve du validateur est plus petite et plus simple.
  - Adapté à d'autres algorithmes d'allocation de registres.
  - Utilisée dans CompCert depuis la version 2.0 (juin 2013).

Conclusion

Comparaison en vraie grandeur de deux méthodologies de vérification formelle → Intérêt de la validation *a posteriori*

Début de bibliothèque de graphes en Coq

Implémentation de référence d'un algorithme connu

Validation *a posteriori* de résultats de solveurs (PLNE) externes