1 Interactions entre passes

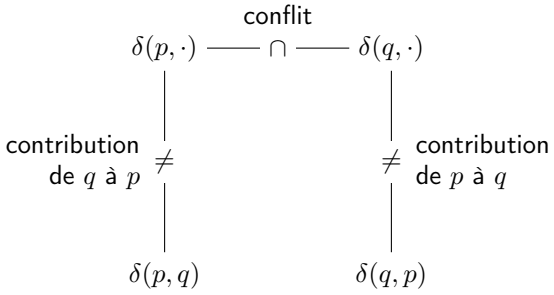
Programmes et diffs

Soient $p_1 \dots p_n$ des optimisations et π un programme. Pour abréger, on note $p_n \dots p_1(\pi)$ (comme un mot) le sortie de la séquence d'optis sur π, c'est-à-dire $(p_n \circ \cdots \circ p_1)(\pi)$. S'il n'y a pas d'ambiguité alors π est omis.

Pour deux séquences d'optis s_1 and s_2, on note $\delta(s_1, s_2)$ le diff entre $s_2(\pi)$ et $s_1 s_2(\pi)$, c'est-à-dire le diff produit par l'action de s_1 sur le programme déjà optimisé par s_2. On note $\cdot$ la séquence vide.

Interactions

On veut alors comparer des diffs pour savoir si deux optimizations p et q interagissent sur le même programme. On définit deux types d'interactions:



Contributions

Une *contribution* de q à p se produit si q génère du code qui est ensuite remanié par p. Cela revient à dire que p effectue sur $q(\pi)$ des modifications qu'il n'effectuait pas sur π.

Pour évaluer ce prédicat, il faut renuméroter les position de $\delta(p, \cdot)$, qui sont indexées sur π, car celles de $\delta(p, q)$ sont indexées sur $q(\pi)$. Il faut donc déplacer les numéros de positions de $\delta(p, \cdot)$ suivant les insertions et suppressions faites par q. Cette opération s'appelle *rebasage* et est là aussi courante dans les gestionnaires de version. Elle n'est définie de façon unique que si p et q ne sont pas en conflit.

Il convient de noter qu'une partie des modifications de $\delta(p, \cdot)$ pourrait ne pas être présente dans $\delta(p, q)$ si q a supprimé les positions où elles se produisent. Toutefois cela serait un conflit ; on a donc nécessairement tout $\delta(p, \cdot)$ dans $\delta(p, q)$. La contribution est donc notée $\delta(p, \cdot) \neq \delta(p, q)$.

Lien avec la commutativité

On dit que p et q commutent si $pq(\pi) = qp(\pi)$. Cela revient à

$$\delta(p, \cdot) \circ \delta(q, p) = \delta(q, \cdot) \circ \delta(q, p).$$

S'il n'y a ni conflit ni contribution entre p et q, alors elles commutent. En effet l'absence de contribution revient à avoir $\delta(p, q) = \delta(p, \cdot)$ et $\delta(q, p) = \delta(q, \cdot)$. La commutativité se ramène donc à

$$\delta(p, \cdot) \circ \delta(q, \cdot) = \delta(q, \cdot) \circ \delta(p, \cdot).$$

Or il n'y a pas de conflit entre p et q, donc la fusion des deux diffs est définie de façon unique. La composition commute donc, d'où l'égalité.

Il y a des situations où q et q commutent mais sont en conflit, par exemple avec le système de réécriture :

$$p(a) = b \qquad q(a) = b$$

Ce genre de commutation ne montre pas une indépendance entre p et q, au contraire. On considère donc ça comme une interaction, ce qui de toute façon ne fait que surestimer la quantité d'interactions. C'est sound.

2 Description des algorithmes de llvm-diff

Tout est dans `/llvm/tools/llvm-diff/DifferenceEngine.cpp`.

Unification triviale de deux blocs : `diff(BasickBlock *L, BasicBlock *R)`.

- Prendre toutes les instructions dans l'ordre et les unifier une par une. Les matchings entres les instructions sont stockés dans `TentativeValues`.
- Si l'unification progresse jusqu'à la fin du bloc, on unifie tout pour de bon.
- Si des instructions diffèrent avant la fin du bloc, `TentativeValues` est vidé et on passe sur l'unification non triviale.

Unification non triviale de deux blocs : `runBlockDiff()`.

- Calcule la distance de Levenshtein entre les deux blocs. Une suppression correspond à un `DC_Left` tandis qu'une insertion correspond à un `DC_Right`. Ces deux opérations ont un coût fixé à 2.
- Unifie de façon gloutonne et laisse `TentativeValues` se remplir pour chaque match trouvé, et s'autoriser à unifier la même instruction plusieurs fois.

  FIXME Cela unifie des valeurs pas nécessairement identifiables et crée des erreurs d'`assert()` lorsque le chemin est déroulé.
- Déroule le chemin et renvoie un optimal en distance de Levensthein sur l'ensemble des renommages... sauf si une erreur se produit.

3 Définitions uniques

Représentation intermédiaire LLVM

On s'intéresse uniquement à des fonctions. La représentation intermédiaire d'une fonction est vue comme un graphe de flot de contrôle dont les nœuds sont des blocs basiques identifiés par un nom, et dont les contenus sont comparés à α-équivalence près.

On formalise l'idée d'un bloc β par un ensemble de lignes muni d'un ordre total $\leq_\beta$ donnant l'ordre des lignes. (C'est plus souple dit comme ça qu'avec une séquence.)

Format du diff

Le diff de deux programmes a deux niveaux : celui des blocs basiques et celui du code à l'intérieur d'un bloc basique commun. Il résulte d'une unification entre la structure du graphe de flot de contrôle, et est composé de suppressions, ajouts, et éditions de blocs basiques.

Le type des diffs est défini par

$$D \equiv \{\text{Remove } \beta \mid \text{Add} \mid \text{Diff } \beta\ \delta,\ \ldots\}.$$

FIXME Les contenus des blocs insérés ou supprimés ne sont pas retenus. Les blocs insérés sont toujours considérés comme ayant une identité nouvelle et jamais unifiés entre deux diffs car il faudrait étendre l'algorithme. Seuls les anciens noms des blocs supprimés et modifiés compte.

Pour chaque bloc modifié, on adjoint un diff de son contenu, qui se calcule avec une liste d'intervalles maximaux soit communs, soit présents uniquement à gauche (supprimés par le patch), soit présents uniquement à droite (ajoutés par le patch). Cette liste est choisie de façon à maximiser le nombre de lignes communes.

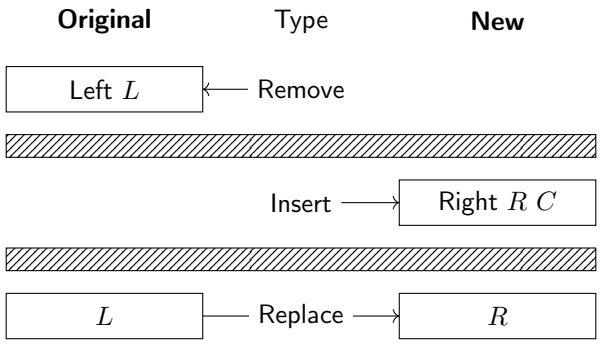


Figure 1: Extrait de la comparaison de deux programmes.

Le diff contient alors l'ensemble des blocs Left, pour lesquels on indique les ligne supprimées, des blocs Right, pour lesquels on indique les lignes insérées et leur position, et des blocs Replace, avec les anciennes et nouvelles lignes. La position d'un bloc Right est notée par le contexte (ou «coupure») C qui est l'ensemble des lignes du programme source qui précèdent le bloc inséré.

Le type des diffs de blocs est noté B et se restreint aux intervalles pris maximaux dans le bloc source β : on note B_{max} le prédicat de maximalité.

$$B \equiv \{\delta \in (\text{Left } L \mid \text{Right } R\ C \mid \text{Replace } L\ R)^+ : B_{\mathsf{max}}(\beta, \delta)\}$$

Proposition. *Pour un ensemble de lignes communes fixé, le diff d'un bloc est unique.*

Soient β_π et β_ρ deux blocs, et $A \subseteq \beta_\pi \cap \beta_\rho$. Il existe une unique façon de partitionner A en intervalles maximaux dans β_π et β_ρ. (La clé est que deux intervalles sont fusionnables ssi ils sont adjacents dans les deux programmes.) On note encore $A = \{A_1 \dots A_n\}$ cette partition, ordonnée par l'ordre commun.

Soit $0 \leq i \leq n$, on note L_i l'intervalle de β_π situé entre A_i et A_{i+1}, et R_i celui de β_ρ. Le diff doit alors être

$$\{f(L_i, R_i, C_i) : L_i \cup R_i \neq \varnothing\}$$

avec

$$f(L, R, C) = \begin{cases} \text{Left } L & \text{si } R = \varnothing \\ \text{Right } R\ C & \text{si } L = \varnothing \\ \text{Replace } L\ R & \text{sinon} \end{cases}$$

où $C_i = \bigcup_{j<i} A_j$.

□

On note old(δ) et new(δ) l'ensemble des intervalles retirés et ajoutés produits par δ :

$$\text{old}(\delta) = \{L : \text{Replace } L\ R \in \delta \vee \text{Left } L \in \delta\}$$
$$\text{new}(\delta) = \{R : \text{Replace } L\ R \in \delta \vee \text{Right } R\ C \in \delta\}$$

Application d'un patch

Étant donné un bloc β et un diff δ, on dit que β' est un patch de β par δ si:

(P_1) L'ordre de β est préservé dans β' :

$$(\leq_{\beta'})_{|\beta \cap \beta'} = (\leq_\beta)_{|\beta \cap \beta'}.$$

(P_2) Les intervalles supprimées par δ sont absents de β' :

$$\forall L \in \text{old}(\delta), \beta' \cap L = \varnothing.$$

(P_3) Les nouveaux intervalles de δ sont des intervalles de β' :

$$\forall R \in \text{new}(\delta), R \text{ est un intervalle pour } \leq_{\beta'}.$$

(P_4) La position des lignes insérées par δ par rapport à β est préservée :

$$\forall(\text{Right } R\ C) \in \delta, \{\ell \in \beta' : \ell \leq_{\beta'} R) = C \cap \beta'.$$

(P_5) La position des lignes remplacées est préservée :

$$\forall(\text{Replace } L\ R) \in \delta, \forall \ell \in \beta \cap \beta', (\ell \leq_\beta L \iff \ell \leq_{\beta'} R).$$

Proposition. *Il existe un unique patch de β par δ ayant pour ensemble de lignes*

$$\beta' = \beta \setminus \bigcup \mathsf{old}(\delta) \cup \bigcup \mathsf{new}(\delta).$$

L'ensemble des lignes est fixé, l'ordre des éléments de β dans β' doit être égal à celui de β, et tous les intervalles insérés doivent rester des intervalles.

Seule la position des blocs insérés peut varier, or (P_4) et (P_5) garantissent leur unicité. En effet, l'unique ligne après laquelle il est légal d'insérer Right R C est

$$\mathsf{max}_{\beta'} \ (C \cap \beta').$$

et l'unique ligne après laquelle il est légal d'insérer Replace L R est

$$\mathsf{max}_{\beta'} \ \{\ell \in \beta \cap \beta' : \ell \leq_\beta L\}.$$

□

FIXME Congruence de diffs

Il faudrait unifier les blocs ajoutés dans chacun des deux diffs pour vérifier que les contenus correspondent. Mais c'est tordu car ça dépend de si les blocs ajoutés ont la même position dans le CFG... et peut-être de s'ils le sont par les mêmes instructions.

Conflit

Deux diffs Δ_1 et Δ_2 sont en conflit si :

(i) L'un d'entre eux supprime un bloc que l'autre modifie, ou

(ii) Les deux induisent sur le même bloc β des diffs δ_1 et δ_2 qui sont en conflit.

Deux diffs d'un même bloc δ_1 et δ_2 sont en conflit si :

(i) L'un deux remplace un bloc L et l'autre supprime une partie de L :

$$\mathsf{Replace}\ L\ _ \in \delta_1 \wedge L \cap \bigcup \mathsf{old}(\delta_2) \neq \varnothing, \text{ou}$$

(ii) L'un deux remplace un bloc L et l'autre insère à l'intérieur de L :

$$\mathsf{Replace}\ L\ _ \in \delta_1 \wedge \mathsf{Right}\ R\ C \in \delta_2 \wedge C \cap L \notin \{\varnothing, L\}, \text{ou}$$

(iii) Les deux insèrent au même endroit après suppression :

$$\mathsf{Replace}\ _\ \ell_1 \in \delta_1 \wedge \mathsf{Replace}\ _\ \ell_2 \in \delta_2 \wedge \forall \ell \in \beta \cap \beta', (\ell \leq_\beta \ell_1 \iff \ell \leq_\beta \ell_2).$$

Union de patchs sur un bloc

Soit β un bloc, et δ_1 et δ_2 deux patchs sur β. On note β' l'ensemble

$$\beta \setminus \bigcup \mathsf{old}(\delta_1 \cup \delta_2) \cup \bigcup \mathsf{new}(\delta_1 \cup \delta_2)$$

qui est l'ensemble des lignes que l'on veut voir apparaître dans le résultat d'une fusion.

Proposition. *Si δ_1 et δ_2 ne sont pas en conflit alors il existe un unique patch de β par δ_1 et δ_2 ayant pour ensemble de lignes β'.*

FIXME

□

FIXME Rebasage et contributions

Très casse-pieds à cause du fait que les deux sources n'ont pas les mêmes noms d'instructions. Nécessite la congruence.