T2: Développement du front-end C++ - Franck

- Tâche 2: Version synchrone
- plugin clang pour remplacer float/double par un autre *type*
- Version 1: *type* n'a aucun destructeur = Cadna
  - Gérer le changement de taille à l'intérieur du pluginn
  - malloc(sizeof(double)) fonctionne, malloc(64) échoue
  - **Instrumentation** avec Cadna concluante
- Version 2: *type* a un destructeur non trivial = FLDLib
  - Changer float par un "fake" type ayant un destructeur: clang++ --include
  - Changer type et toutes les operations type:: par celles de *type*
- Besoin d'un fichier json qui determine le périmètre de l'instrumentation

T2: Développement du front-end Unisim - Yves

- Version synchrone

- Émulation type bochs
- Facilité d’expérimentation
  - Mémoire shadow
  - Stocker un graphe de contrôle pour gérer les tests instables ?
- Jeux d’instructions
- virtio

T2: Mémoires Shadow - Julien

- Mémoires shadow d'E-ACSL

- Librairie LGPL v2, écrite en C, disponible sous linux, mac, windows
- Compétitive avec celle de clang
- Adaptation au contexte InterFlop
  - Différents front-ends, différents back-end
  - Les métas-info liés aux flottants (variables, mémoires et registres)
  - Conserver ces infos lorsqu'elles sont transportées par des registres entiers

  - Allocation dynamique dans les méta-infos (ex: précision symbolique)
  - Informations attachées au contrôle (ex: point de synchronisation des tests instables)

Tâche 3: New models for error estimation and composite analysis

Franck Védrine, Yves Lhuillier, Julien Signoles

Tâche 3: Objectifs – M12-M48

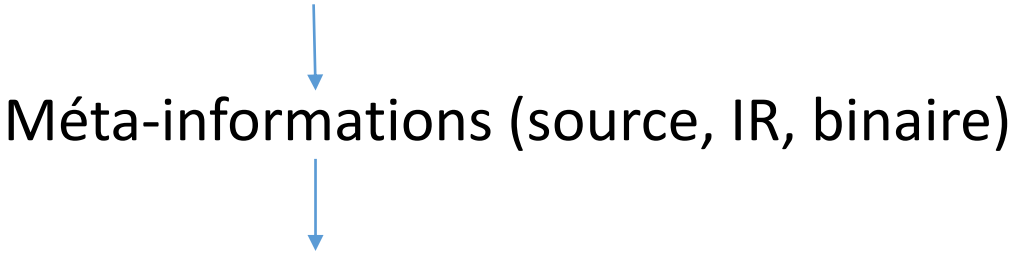
- Localisation des instabilités
  - delta-debugging
  - génération de traces, origines des principales instabilités et analyse arrière
- Permettre des changements dynamiques de mode/type d'analyse
  - analyse stochastique asynchrone $\Rightarrow$ analyse formelle $\Rightarrow$ analyse synchrone $\Rightarrow$ analyse stochastique asynchrone
- Garantir la robustesse, l'absence d'instabilités sur des parties du code
  - Petite erreur en entrée $\Rightarrow$ petite erreur en sortie
  - Tests instables
  - Validation formelle et résumé

Sous-tâche 3.1: Infrastructure for error estimation analyses with many back-end

- Définir le découpage pour différents usages
  - Un fichier JSON ? le code exécuté, les points de switch, le code analysé
- Les outils
  - Changement de front-end
    - CERE – découpage niveau LLVM, isolation des boucles à fort calcul
    - nouvelle version
  - Changement de back-end – à définir
    - Une API commune à tous les back-ends, synchrones, asynchrones, multi-exécution, tests instables – avec option pour indiquer si la feature est supportée

Sous-tâche 3.2: Debugging session to set up the analysis switches

- Un changement dynamique d'analyse en deux temps
  - Un temps debug pour ajuster/changer d'analyse pour la mise au point



  - Une analyse automatique et intégrée

- 2 outillages et une interface utilisateur à définir sur l'aspect debug
  - delta-debugging = debugging asynchrone
  - capacité à sauvegarder l'origine + revenir en arrière = debugging synchrone

Tâche 3: avance de phase

- API pour les différents back-ends
- Amélioration des back-ends liés à l'analyse conservatrice
  - Etat de l'art trop sur-approximé – intégrer une analyse symbolique pilotée par des points d'arrêt liés à une détection d'instabilité/sur-approximation
- Expérimentations pour choisir la meilleure combinaison des outils
- Interface utilisateur (voir tâche 5)
  - vraie interface de pilotage - graphiques
  - gdb, mode console ?