Langages et compilation

Jean Privat (UQAM)

17 avril 2024 — Midi-recherche

Qui suis-je ? Que fais-je ?

Qui suis-je ?

2001-2002. DEA (≈ Maîtrise), Université de Montpellier (France)

- « Analyse de types et graphes d'appels en compilation séparée »

2002-2006. Thèse, Université de Montpellier

- « De l'expressivité à l'efficacité, une approche modulaire des langages à objets. Le langage PRM et le compilateur prmc »

2006-2007. Postdoc, Purdue Universily (Indiana, USA)

- OVM une machine virtuelle temps réel pour Java
- StreamFlex un modèle de programmation parallèle temps réel pour Java

Depuis 2007. Professeur UQAM (Québec, Canada)

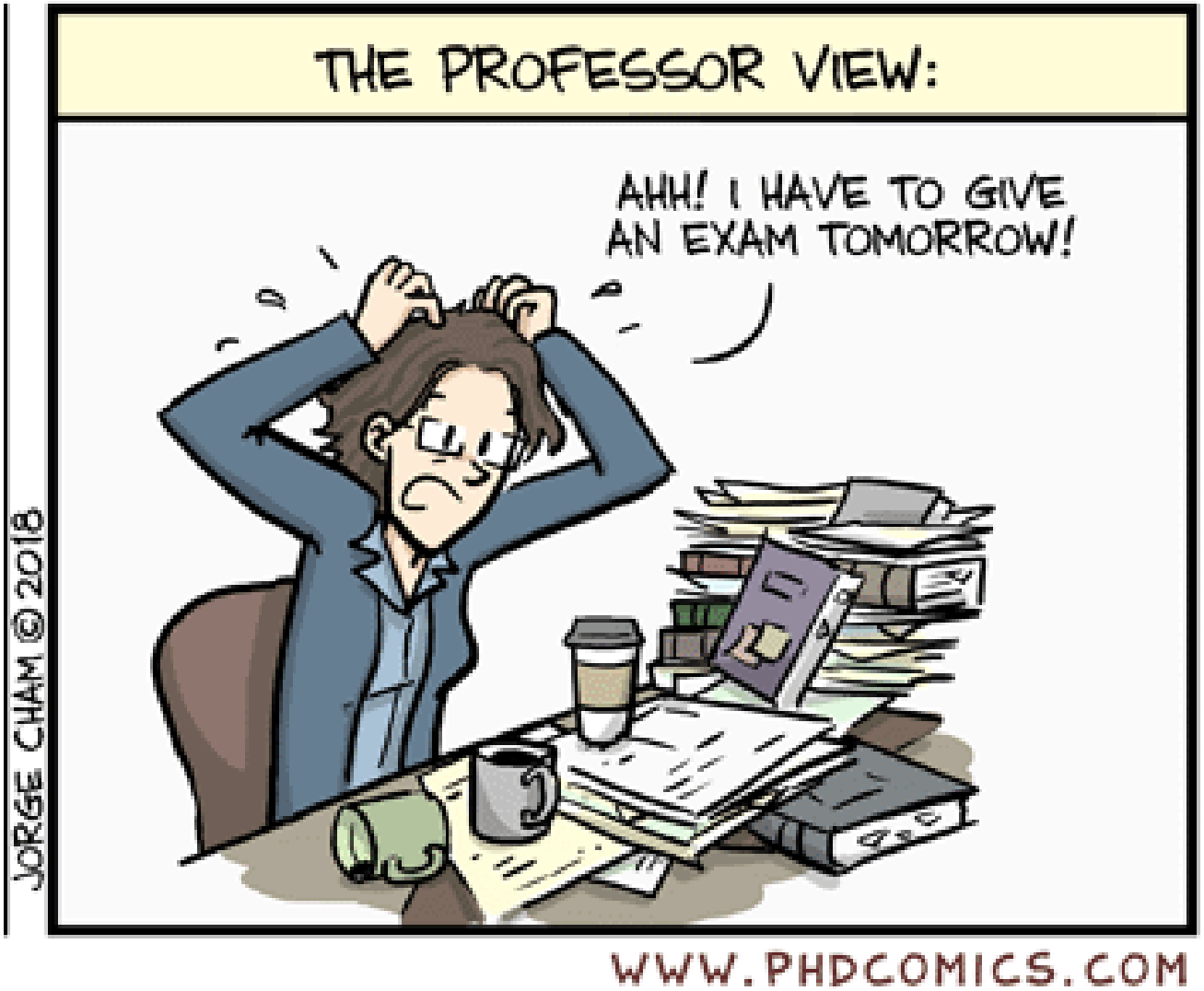
- INF1070 (unix), INF217x (asm), INF317x (os), INF5000 (compil), INF600C (sécu), INF7845 (oop), INF889A (analyse de prog)
- https://privat.uqam.ca

Ce que ma famille pense que je fais

source: Matrix (1999)

Ce que mes collègues pensent que je fais

Ce que je fais



source: phdcomics (2018)

Centres d'intérêts universitaires

Principaux

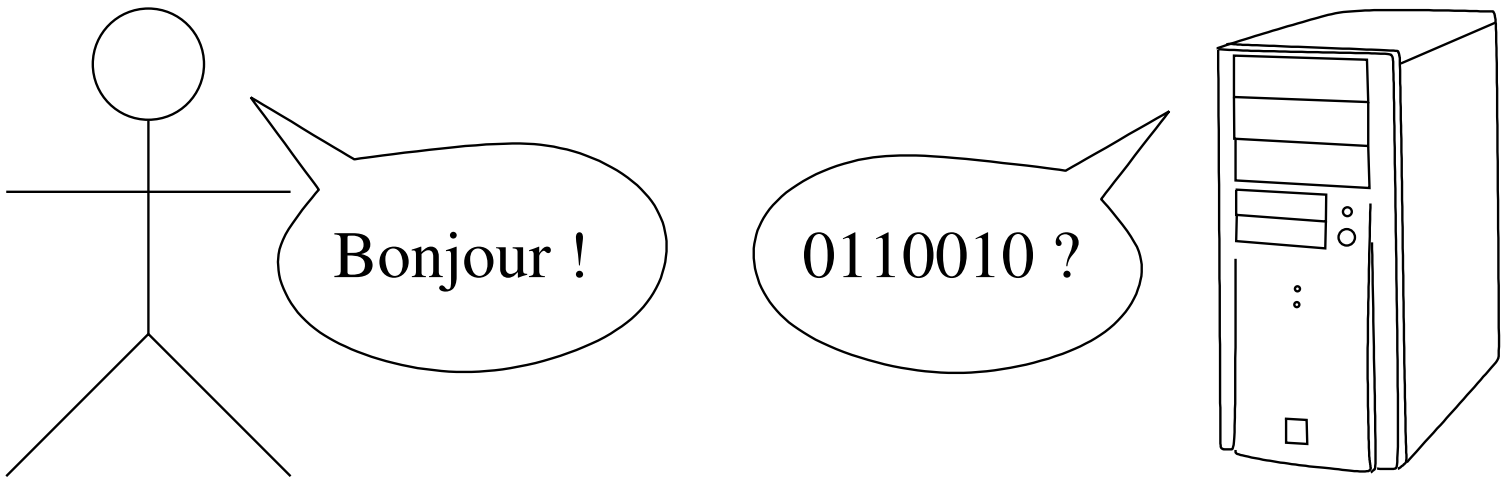
- Programmes
- Bibliothèques
- Systèmes d'exploitation
- Compilateurs, interpréteurs et machines virtuelles
- Langages de programmation

Extra

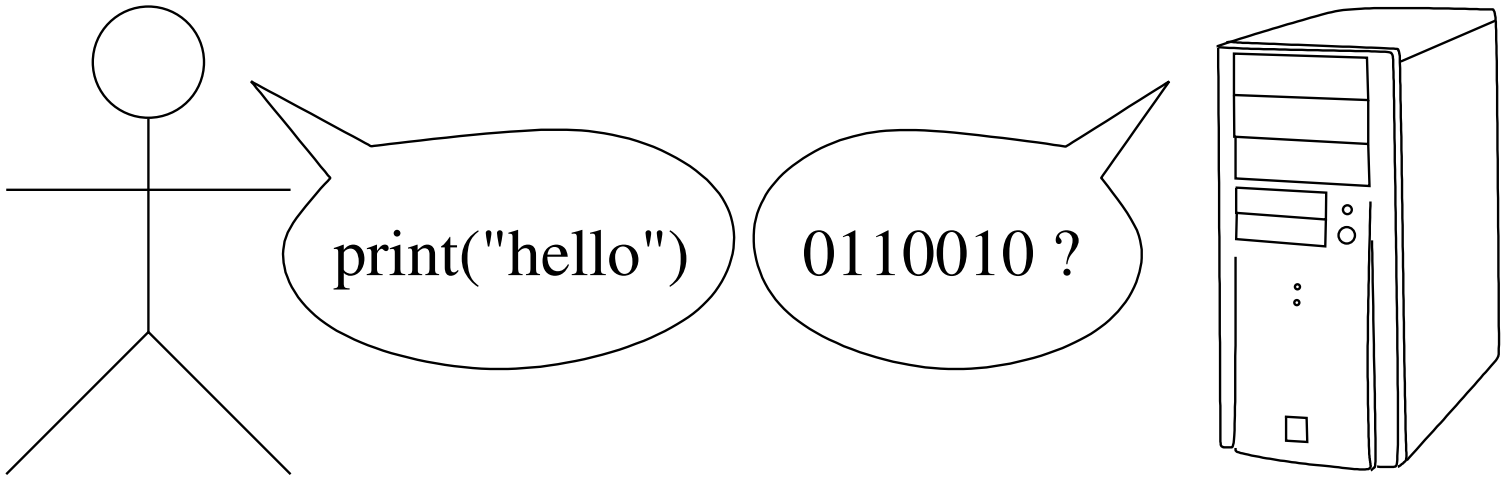
- Programmation orienté objets (conception)
- Cybersécurité applicative
- Logiciels libres
- Pédagogie en informatique

Langages et compilation

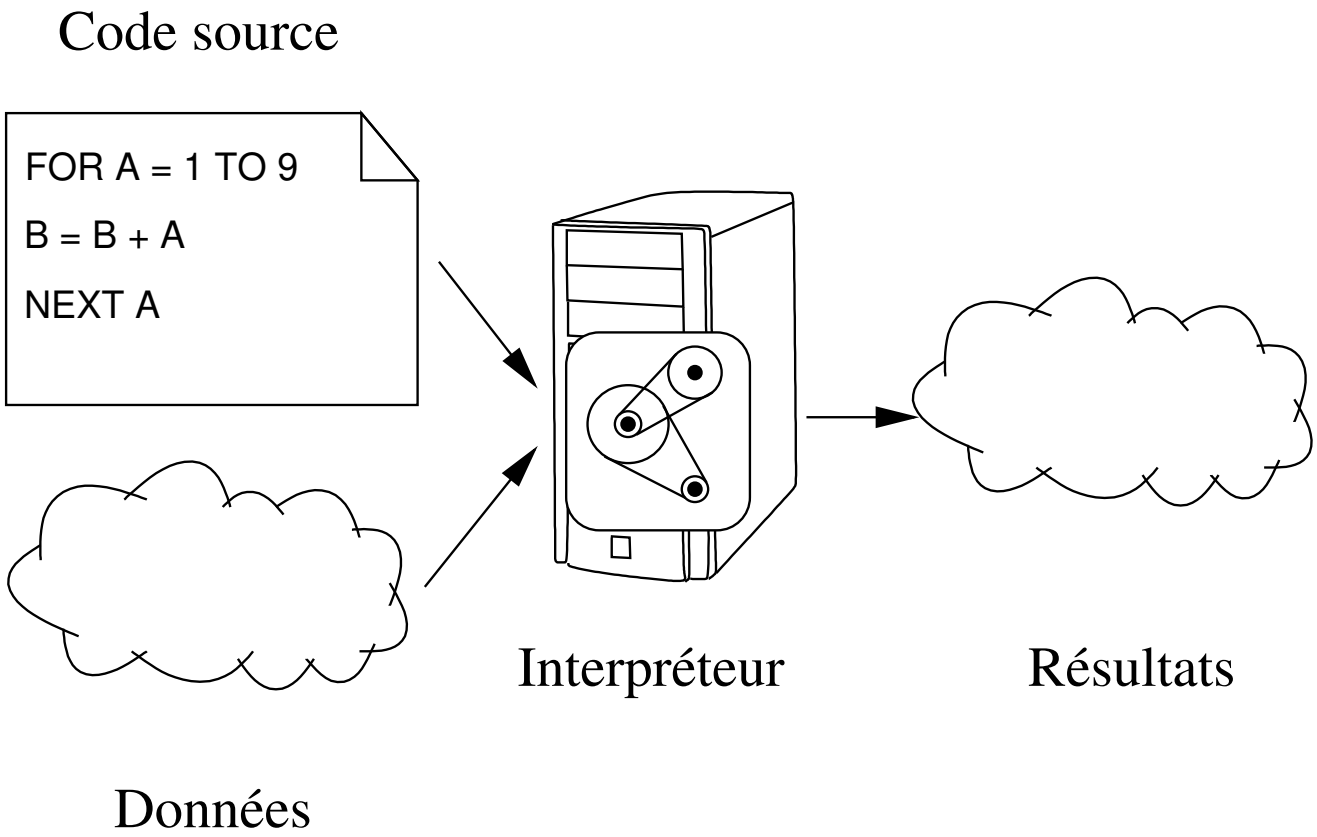
Interface Humain-Machine



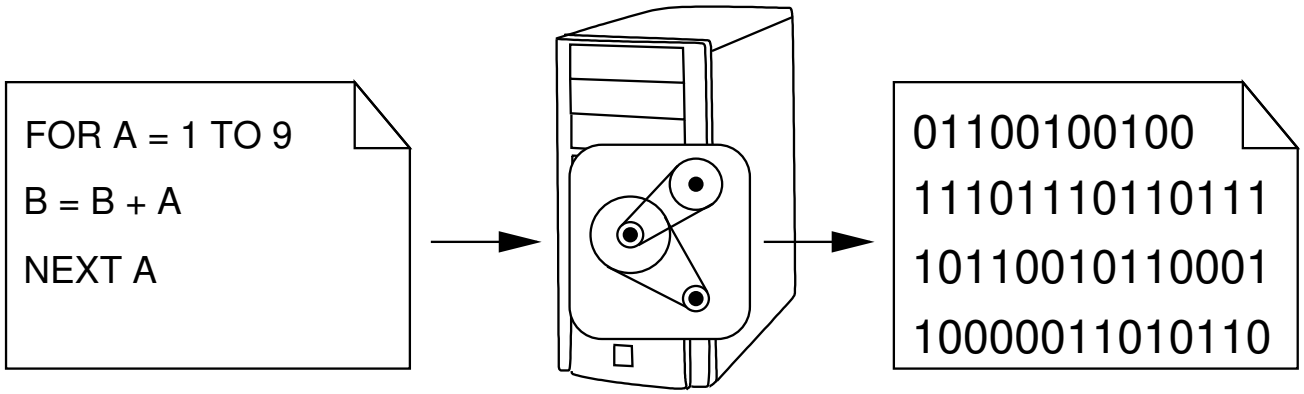
Interface Humain-Machine (mieux?)

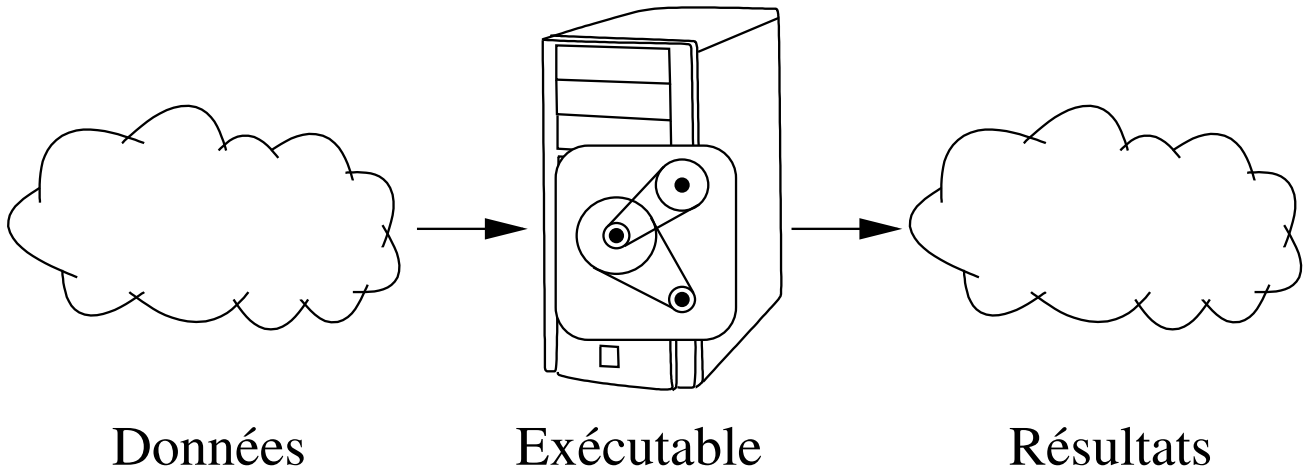


Interpréteur



Compilateur





Projets de recherche

Projets de recherche (et de création)

Programmer des programmes

- C'est amusant

Programmer des compilateur, interpréteur et machines virtuelles

- C'est encore plus amusant
- Mais, c'est pas facile

- Gros et complexe
- Si on veut de la performance et du passage à l'échelle
  - Encore plus gros et complexe

- Difficile à coder et maintenir sans y passer tout son temps
  - Et sans une équipe d'ingénieurs
- Difficile d'entrée pour de nouveaux étudiants (et contributeurs)

Projet Nit

https://nitlanguage.org

- Langage OO “maison”
- Statiquement typé, nombreuses fonctionnalités cool
  - Héritage multiple propre, raffinement de classes, typage adaptatif, etc.
- Compilateur optimisant (et un interpréteur naïf pour prototyper)
- Philosophie de conception : propre, KISS et POLA
- Bac-à-sable : plateforme d’expérimentation et de créativité

Trucs en Nit

- Opportunity : doodle qui respecte la vie privée
- Gamnit : cadriciel de jeux vidéo

Projet Pharo

https://pharo.org

- Dialecte moderne de Smalltalk, avec une communauté internationale
- Orienté-objet, dynamiquement type, hautement réflexif, “live coding”
  - Système d'exploitation + langage de programmation + IDE
- Machine virtuelle avec compilateur jute-à-temps
- Philosophie de conception: minimal, puissant, réflexif
- Mis en production : Niche, mais utilisé par quelques industriels

Trucs en Pharo

- Roassal : bibliothèque de visualisation

La recherche en informatique à l'UQAM

Midi-Recherche

On invite des enseignants-chercheurs, qui présentent, illustrent et font la bande annonce de

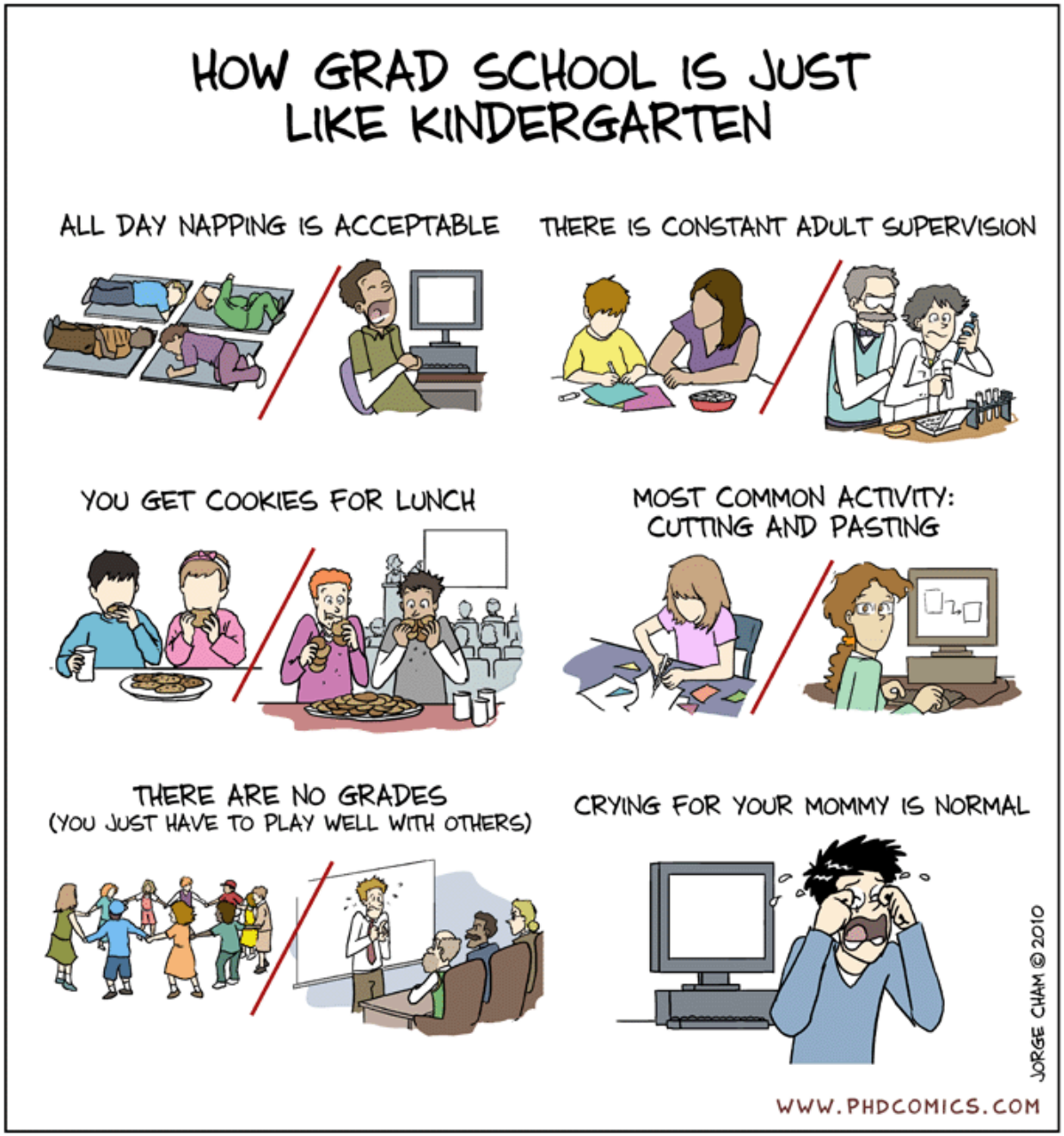
- Leur domaine de recherche
- Les *grandes* questions de recherche
- Leurs travaux de recherche en cours
- Leurs travaux de recherche passé
- Les défis de recherche futurs

Et c'est vraiment intéressant (j'ai presque assisté à tous)

Étudiante-chercheuse, étudiant-chercheur

Question peu abordée

- Quel est le rôle et/ou la place des personnes étudiantes ?
- Quelle est la journée type ?
- En vrai ?



source : phdcomic (2010)

Qui sont les personnes étudiantes-chercheuses ?

1er cycle : principalement BIGL

- Bourse de recherche de premier cycle (BRPC)
- Profile *honor*, cours-projet individuel INF6200
- Stage COOP (ou non COOP)

2e cycle : Maîtrise en informatique (dite recherche)

- 5 cours avancés (et un de méthodologie)
- Un projet de recherche (27 crédits)

3e cycle : Doctorat en informatique

- *New game+*
- 2 cours (banque de la maîtrise)
- Un gros projet de recherche (74 crédits)

Formation **par** la recherche

- Problèmes parfois ouverts et souvent variés
- Méthodologie scientifique (ordre et discipline)
- Sur les épaules de géants
- Compréhension globale de la problématique étudiée

Approche de travail différente du travail scolaire ou industriel

- Objectif de moyens, et non de résultats
- Chemin tout autant intéressant que la destination
- On essaye des trucs. . .

Extensions de langages

Extensions de langages

- Étendre un langage de programmation
  - Quoi ? Pourquoi ? État de l'art ? Alternatives ? Justifications ? Compromis ? etc.
  - Pourquoi pas une bibliothèque ?
- Syntaxe, sémantique (et messages d'erreur)
- Implémentation : interpréteur et/ou compilateur
- Mise en œuvre, utilisation, validation

Types nullables

- « Prévention de déréférencements de nul », 2012
- Jean-Sébastien Gélinas. Maîtrise en Informatique.
  Co-supervision avec Étienne Gagnon
- Distinguer les types qui acceptent ou refuse la valeur `null`

« La bonne gestion des déréférencements de nul dans les langages à objets statiquement typés est un problème complexe qui est loin d'être nouveau. Plusieurs approches existent, mais aucune n'est parfaite. Les diverses solutions au déréférencement de nul se partagent trois dimensions : (1) la simplicité, (2) l'exactitude des résultats et (3) le typage statique. Présentement, à notre connaissance, aucune solution ne présente ces trois dimensions. Un des problèmes majeurs de la gestion des déréférencements de nul est la gestion des attributs, notamment lors de l'instanciation des objets. L'approche que nous présentons ici pour gérer les déréférencements de nul est une approche (1) simple et (2) exacte. Bien que notre approche présente un composant de typage statique, nous ajoutons aussi des tests dynamiques pour la compléter, ce qui fait qu'elle ne respecte pas totalement les trois dimensions. Nous introduisons aussi des analyses statiques pour détecter l'initialisation d'attributs dans les constructeurs pour ensuite, à l'aide d'optimisations, supprimer les tests dynamiques que nous avons ajoutés à la phase précédente. [. . .] »

Parallélisme

- « Extensions parallèles pour le langage Nit », 2013
- Sylvain Poirier. Maîtrise en Informatique.
  Co-supervision avec Guy Tremblay
- Proposer un modèle de programmation parallèle basé sur Linda

- « CelluloNit, une implémentation du modèle d'acteur en Nit », 2018
- Romain Chanoir. Maîtrise en Informatique,
  Co-supervision avec Guy Tremblay.

« [. . .] Nous proposons une implémentation du modèle acteur en Nit, appelée CelluloNit, qui bénéficie à la fois de la simplicité de Celluloïd et des avantages procurés par le typage statique. L'exécution d'une série de programmes de tests de performance place CelluloNit entre Akka et Celluloïd en termes de performance. CelluloNit représente donc une première implémentation intéressante du modèle acteur en Nit. »

Programmation polyglotte

- « Interface native de Nit, un langage de programmation à objets », 2012
- Alexis Laferrière. Maîtrise en Informatique.
- Proposition d'un modèle de programmation pour appeler des fonctions C (dans les deux sens)
  - Accès aux services du système d'exploitation
  - Accès à des bibliothèques tierces

- « Applications mobiles portables et de haute qualité : du prototype à la ligne de produits par le raffinement de classes et la programmation polyglotte », 2018
- Alexis Laferrière. Doctorat en Informatique.

« [. . .] Cette thèse introduit une nouvelle solution pour réaliser des applications portables et pour les adapter à Android et à iOS. Notre solution combine trois approches. (i) Des API portables servent à réaliser un prototype rapidement. (ii) Une organisation en ligne de produits, réalisée par le raffinement de classes, permet une évolution graduelle du prototype en deux applications adaptées à Android et à iOS. (iii) La programmation polyglotte, via l'interface de fonctions étrangères (FFI) de Nit, donne accès aux API natives dans leur entièreté et leur langage natif. [. . .] »

Programmation polyglotte

Stage 1er cycle

- « Applications Android de haute qualité écrites en Nit », 2014
- Frédéric Vachon, Stagiaire CRSNG.
- `jwrapper` pour générer des classes externes Nit associées à des classes Java
  - Simplifie la liaison avec les API Android

Démo

- Calculatrice
- Gamnit

Généricité covariante

- « Implémentation homogène de la généricité covariante dans Nit, un langage à objets en héritage multiple », 2014
- Alexandre Terrasa. Maîtrise en Informatique.
- Maintenir efficacement l'information de type à l'exécution

« [. . .] La politique covariante autorise une plus grande expressivité dans l'utilisation des types génériques. En contrepartie, elle nécessite un mécanisme de résolution des types génériques ouverts et elle entraîne un problème d'insécurité dans les appels de méthodes utilisant des types génériques redéfinis de manière covariante. L'implémentation homogène permet de partager la même implémentation du corps des méthodes entre les variations génériques d'une même classe. Le coût en mémoire de la généricité est alors réduit au strict minimum. En revanche, cette approche pose les problèmes de représentation des types génériques, d'implémentation du test de sous-typage et d'implémentation du mécanisme de résolution. Nous présentons une solution basée sur l'utilisation de descripteurs de types et de tables de résolution et nous l'implémentons dans un compilateur pour le langage Nit [. . .] »

Sélection multiple

- « Implémentation des multiméthodes en Nit », en cours
- Hugo Leblanc, Maîtrise en Informatique
- Le receveur n'est plus seul à faire le polymorphisme
  - Tous les arguments participent
  - C'est très UQAMien
  - C'est aussi plus difficile que ça en a l'air

Chaînes de caractères

- « Des chaînes de caractères efficaces et résistantes au passage à l'échelle, une proposition de modélisation pour les langages de programmation à objets »
- Lucas Bajolet, Maîtrise en Informatique, 2017
- Traitement des chaines, unicode, et tas d'octets

« [. . .] Malgré son importance, peu d'études se sont penchées sur ce pan de l'informatique et l'ensemble des techniques aujourd'hui utilisées ont peu évolué durant les trente dernières années. Dans cette étude, nous étudions les implémentations actuelles des chaînes dans les langages de programmation, tant au niveau des structures de données que des codages. Nous développons un modèle objet de représentation des chaînes de caractères efficace et résistant au passage à l'échelle, reposant sur une combinaison de cordes et chaines plates, entièrement implémenté dans le langage Nit. Cette combinaison nous permet de combler les problèmes connus des chaînes de caractères telles que représentées dans les langages passés et actuels [. . .] »

- En fait, c'est pas de l'extension de langage, mais le compilateur a été bricolé

Autodocumentation

- « Auto-documentation assistée de logiciels: génération et maintenance de fichiers README avec l'outil nitreadme », 2019
- Alexandre Terrasa, Doctorat en Informatique.
  Co-supervision avec Guy Tremblay

« [. . .] Nous proposons donc une approche permettant d'assister !'écrivain de fichiers README, tant durant la rédaction que la maintenance. Lors de la rédaction, notre approche consiste à suggérer des cartes de documentation, i.e., des extraits de documentation produits par le générateur de documentation d'API pouvant être importés directement dans le corps du README. Puis, durant la maintenance, notre approche assure que le contenu de ces cartes soit tenu synchronisé avec le code source. La mise en œuvre de notre approche repose sur l'alignement du contenu du fichier README avec le contenu de l'API qu'il documente, c'est-à-dire l'établissement de correspondances entre les éléments clés du README et les entités du code source [. . .] »

- Système de méta-inspection, qui fait tourner nitweb

Programmation par contrat

- « Une implémentation de la programmation par contrat en Nit, un langage à objet », 2020
- Florian Deljarry. Maîtrise en Informatique
- Validation dynamique de contraintes métier (super-assertions configurables)

« Le paradigme de programmation par contrat, ou plus connu sous le nom de design by contrat™ en anglais, se base sur le concept de contrat pour venir renforcer la fiabilité des systèmes logiciels. L'idée derrière cette méthodologie est de fournir au programmeur un moyen pour définir de façon programmative les obligations des différents éléments du langage telles que les méthodes, interfaces et classes afin de vérifier l'adéquation de l'implémentation envers sa spécification. [...] Contrairement aux approches existantes, notre proposition met en œuvre une représentation basée sur l'utilisation de plusieurs points d'entrée. Cette approche permet ainsi de faire coexister plusieurs versions d'une même méthode avec un niveau de vérification différent. Le principal avantage est la possibilité de pouvoir déterminer statiquement la vérification nécessaire pour chaque site d'appel, permettant ainsi de diminuer l'impact dynamique. [...] »

Sécurité applicative

Cybersécurité

- La cybersécurité, c'est amusant
  - Surtout quand c'est la faute du programmeur
- « Sécurité applicative »
  - On s'intéresse aux défauts de sécurité dans les logiciels

Analyse de programmes

- Voir la présentation du professeur Quentin Stiévenart
- 8 novembre 2023 « Analyse de programmes »

Analyse de programmes assembleur

- « Correction automatique de travaux binaires : De la découverte de code à la génération de tests, une approche autonome »
- Philippe Pépos Petitclerc. Maîtrise en Informatique
- Prendre un TP d'étudiant, prendre la solution de l'enseignant, identifier où le comportement diverge, produire un exemple.

« [. . .] Nous mettons l'accent sur les techniques modernes d'exécution symbolique de programmes binaires. Nous nous penchons également sur les méthodes de génération automatique de tests. En se basant sur ces recherches, nous proposons une stratégie de génération de suites de tests couvrantes sans accès au code source du programme testé ni à sa spécification. Nous validons l'efficacité de la stratégie proposée en l'implémentant dans l'optique de la correction automatique de travaux universitaires soumis par des étudiants [. . .] »

En cours

Analyse de JS

- « Analyse de programmes JavaScript pour la sécurité »
- Olivier Arteau, Maîtrise en Informatique

Évasion d'antivirus

- « Évasion de systèmes de sécurité logiciels par identification de fragilités et transformations automatiques »
- Philippe Pepos Petitclerc, Doctorat en Informatique

Auto-exploitation

- « Exploitation automatique de vulnérabilité de désérialisation en Java »
- Philippe Grégoire. Maîtrise en Informatique

En cours

Anti-fonctionalités

- « Détection des comportements insidieux dans les applications mobiles pour enfants »
- Emmanuel Merlo, Maîtrise en Informatique,
  Co-direction avec Maude Bonenfant (dept de comm)

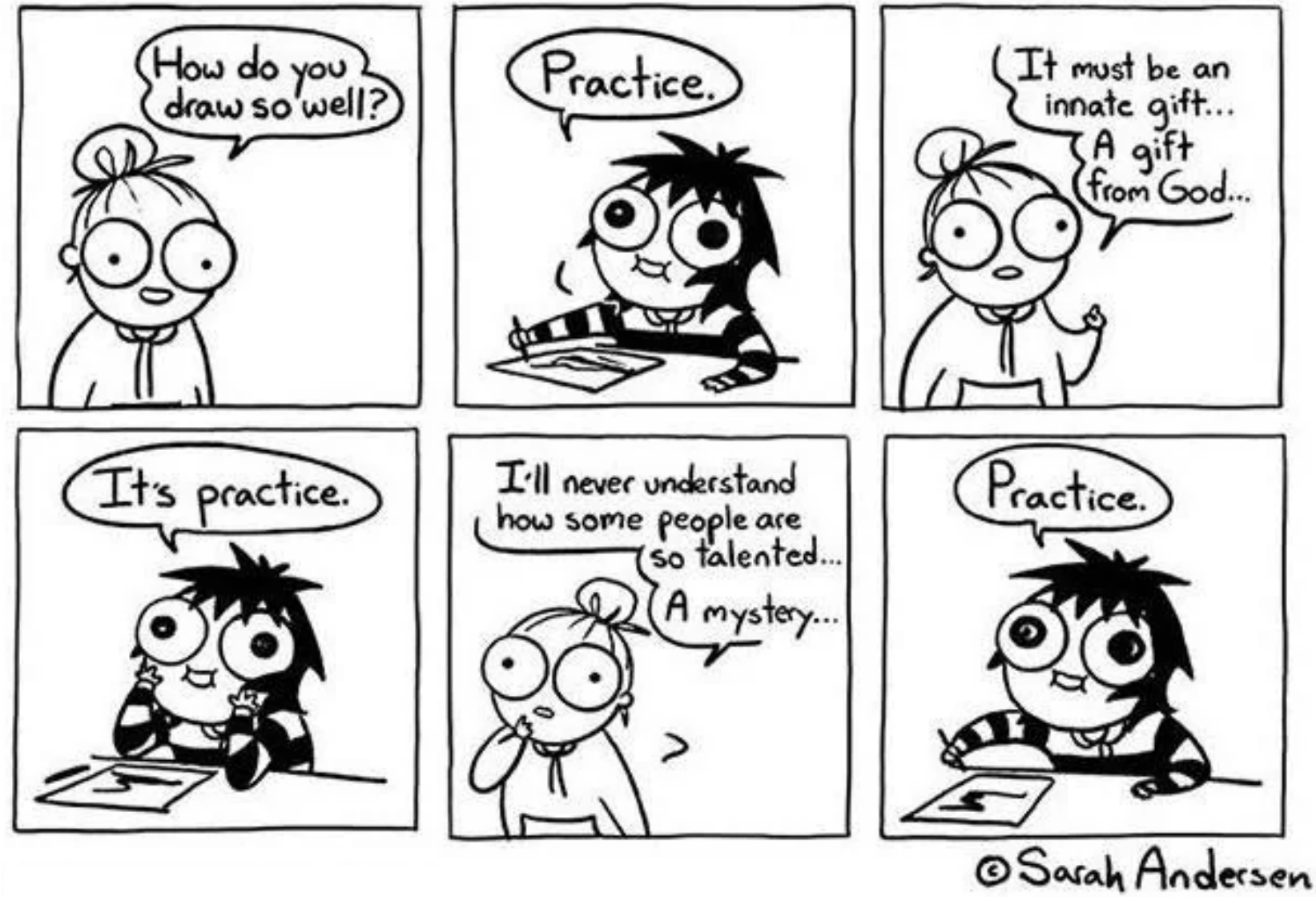
Vie privée

- « Analyses d'impact relatives à la protection des données dans un contexte d'identité numérique »
- Nariman Foughali, Maîtrise en Informatique,
  Co-direction avec Sébastien Gambs

Merci

Questions ?

git gud



source : sarah's scribbles (2017)