Vers une implémentation automatique de fonctions `libm`

Présentation EVA-Flo

Sylvain Chevillard
Christoph Quirin Lauter

Arénaire team
Laboratoire de l'Informatique et du Parallélisme
École Normale Supérieure de Lyon

Lyon, Perpignan, 5 et 18 octobre


Histoire du développement de fonctions `libm`

Développement en Arénaire – 1

Première fonction dans `crlibm`

Développement en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour

Développement en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes d'approximation

Développement en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes d'approximation
- code `C` portable
- deuxième étape basée sur une bibliothèque flottante écrite à cet effet

Développement en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes d'approximation
- code `C` portable
- deuxième étape basée sur une bibliothèque flottante écrite à cet effet
- preuve écrite à la main, très complexe

Développement en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes d'approximation
- code `C` portable
- deuxième étape basée sur une bibliothèque flottante écrite à cet effet
- preuve écrite à la main, très complexe
- temps nécessaire : une thèse

Développement en Arénaire members – 2

Implémentation alternative

Développement en Arénaire members – 2

Implémentation alternative

- `exp(x)` par Christoph

Développement en Arénaire members – 2

Implémentation alternative

- `exp(x)` par Christoph
- arrondi correct en une étape

Développement en Arénaire members – 2

Implémentation alternative

- `exp(x)` par Christoph
- arrondi correct en une étape
- utilisation d'unités spéciales Itanium, code assembleur

Développement en Arénaire members – 2

Implémentation alternative

- `exp(x)` par Christoph
- arrondi correct en une étape
- utilisation d'unités spéciales Itanium, code assembleur
- preuve écrite à la main, très complexe, fausse

Développement en Arénaire members – 2

Implémentation alternative

- `exp(x)` par Christoph
- arrondi correct en une étape
- utilisation d'unités spéciales Itanium, code assembleur
- preuve écrite à la main, très complexe, fausse
- temps nécessaire : 3 mois de stage de maîtrise

Développement en Arénaire members – 3

D'autres fonctions en `crlibm` : `atan(x)`, `log(x)`...

Développement en Arénaire members – 3

D'autres fonctions en `crlibm` : `atan(x)`, `log(x)`...

- Génération des fichiers avec les coefficients des polynômes d'approximation en Maple

Développement en Arénaire members – 3

D'autres fonctions en `crlibm` : `atan(x)`, `log(x)`...

- Génération des fichiers avec les coefficients des polynômes d'approximation en Maple
- Calcul de la norme infinie de la fonction d'erreur en Maple

Développement en Arénaire members – 3

D'autres fonctions en `crlibm` : `atan(x)`, `log(x)`...

- Génération des fichiers avec les coefficients des polynômes d'approximation en Maple
- Calcul de la norme infinie de la fonction d'erreur en Maple
- Preuve Gappa écrites à la main

Développement en Arénaire members – 3

D'autres fonctions en `crlibm` : `atan(x)`, `log(x)`...

- Génération des fichiers avec les coefficients des polynômes d'approximation en Maple
- Calcul de la norme infinie de la fonction d'erreur en Maple
- Preuve Gappa écrites à la main
- temps nécessaire : à peu près 1 mois par fonction

Quel est le problème ?

Pourquoi le processus de développement est-il si lent ?

Quel est le problème ?

Pourquoi le processus de développement est-il si lent ?

En fait, je pensais qu'on effectuait toujours les mêmes tâches...

Automatisation du processus d'implémentation

Histoire du développement de fonctions `libm`

Automatisation du processus d'implémentation

Démonstration...

Conclusions

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve
- Intégrer tout dans un produit final

Étapes du processus d'implémentation

Tâche : implémenter une fonction f sur un domaine $[a, b]$ avec une précision de k bits

- Analyser le comportement de f sur $[a, b]$
- Trouver une réduction d'argument appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve
- Intégrer tout dans un produit final
- Surtout ne plus jamais y toucher parce que c'était tant de boulot

Une chaîne d'outils prototypée – 1

Une chaîne d'outils prototypée automatique pour le processus d'implémentation

Une chaîne d'outils prototypée – 1

Une chaîne d'outils prototypée automatique pour le processus d'implémentation

- Travail d'équipe par
  - S. Chevillard (Polynômes à coefficients flottants)
  - Ch. Lauter (Implémentation du polynôme, preuve, intégration)
  - G. Melquiond (Gappa)
  - et d'autres membres de l'équipe Arénaire

Une chaîne d'outils prototypée – 1

Une chaîne d'outils prototypée automatique pour le processus d'implémentation

- Travail d'équipe par
  - S. Chevillard (Polynômes à coefficients flottants)
  - Ch. Lauter (Implémentation du polynôme, preuve, intégration)
  - G. Melquiond (Gappa)
  - et d'autres membres de l'équipe Arénaire
- Écrite en
  - Pari/GP
  - `C`, `C++`
  - Shell scripts
  - un langage interne : `arenaireplot`

Une chaîne d'outils prototypée – 1

Une chaîne d'outils prototypée automatique pour le processus d'implémentation

- Travail d'équipe par
  - S. Chevillard (Polynômes à coefficients flottants)
  - Ch. Lauter (Implémentation du polynôme, preuve, intégration)
  - G. Melquiond (Gappa)
  - et d'autres membres de l'équipe Arénaire
- Écrite en
  - Pari/GP
  - `C`, `C++`
  - Shell scripts
  - un langage interne : `arenaireplot`
- Ciblée sur
  - une implémentation `C` portable
  - utilisant l'arithmétique double, double-double et triple-double
  - avec une évaluation par schéma de Horner

Une chaîne d'outils prototypée – 2

Étapes automatisées :

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve

Parties qui manquent :

- Analyser le comportement de f sur $[a, b]$

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve

Parties qui manquent :

- Analyser le comportement de f sur $[a, b]$
- Trouver des réduction d'argument avec tabulation etc.

Une chaîne d'outils prototypée – 2

Étapes automatisées :

- Trouver une translation appropriée
- Calculer un polynôme d'approximation p^*
- Déduire de p^* un polynôme p avec coefficients flottants
- Implémenter p dans une arithmétique flottante
- Borner les erreurs d'arrondi, écrire une preuve pour la borne
- Relire la preuve pour trouver des fautes
- Borner l'erreur d'approximation, $\| \frac{p-f}{f} \|_\infty$, écrire la preuve

Parties qui manquent :

- Analyser le comportement de f sur $[a, b]$
- Trouver des réduction d'argument avec tabulation etc.
- Intégration finale

Démonstration...

Démonstration

Task : Implémenter

$$f(x) = e^{\cos x^2 + 1}$$

dans l'intervalle

$$I = [-2^{-8}; 2^{-5}]$$

avec une précision de 64 bits

Démonstration

Task : Implémenter

$$f(x) = e^{\cos x^2 + 1}$$

dans l'intervalle

$$I = [-2^{-8}; 2^{-5}]$$

avec une précision de 64 bits

Essayons...

Conclusions

Histoire du développement de fonctions `libm`

Automatisation du processus d'implémentation

Démonstration...

Conclusions

Nouvelles fonctions dans `crlibm`

Dernières fonctions dans `crlibm`

Nouvelles fonctions dans `crlibm`

Dernières fonctions dans `crlibm`

- `sinpi(x)`, `cospi(x)`, `tanpi(x)`

Nouvelles fonctions dans `crlibm`

Dernières fonctions dans `crlibm`

- `sinpi(x)`, `cospi(x)`, `tanpi(x)`
- arrondi correct en deux étapes

Nouvelles fonctions dans `crlibm`

Dernières fonctions dans `crlibm`

- `sinpi(x)`, `cospi(x)`, `tanpi(x)`
- arrondi correct en deux étapes
- tous les codes d'évaluation générés automatiquement

Nouvelles fonctions dans `crlibm`

Dernières fonctions dans `crlibm`

- `sinpi(x)`, `cospi(x)`, `tanpi(x)`
- arrondi correct en deux étapes
- tous les codes d'évaluation générés automatiquement
- temps nécessaire : 2 jours

Eva-Flo(h) et Adam-Puce

- Le prototype répond à un sous-ensemble très restreint des objectifs d'Eva-Flo (il me semble)

Eva-Flo(h) et Adam-Puce

- Le prototype répond à un sous-ensemble très restreint des objectifs d'Eva-Flo (il me semble)
- Développement de l'outil `arenaireplot` :

Eva-Flo(h) et Adam-Puce

- Le prototype répond à un sous-ensemble très restreint des objectifs d'Eva-Flo (il me semble)
- Développement de l'outil `arenaireplot` :
  - Développement depuis 1 an et demi
  - Certaines choses marchent bien et arrivent à maturité
  - On a commis des fautes de conception difficiles à corriger a posteriori

Eva-Flo(h) et Adam-Puce

- Le prototype répond à un sous-ensemble très restreint des objectifs d'Eva-Flo (il me semble)
- Développement de l'outil `arenaireplot` :
  - Développement depuis 1 an et demi
  - Certaines choses marchent bien et arrivent à maturité
  - On a commis des fautes de conception difficiles à corriger a posteriori
- Le prototype se sert d'outils externes

Eva-Flo(h) et Adam-Puce

- Le prototype répond à un sous-ensemble très restreint des objectifs d'Eva-Flo (il me semble)
- Développement de l'outil `arenaireplot` :
  - Développement depuis 1 an et demi
  - Certaines choses marchent bien et arrivent à maturité
  - On a commis des fautes de conception difficiles à corriger a posteriori
- Le prototype se sert d'outils externes
  - Que faire quand l'outil externe ne marche pas ?
  - Communication pénible par fichiers ou arguments en ligne de commande

Eva-Flo(h) et Adam-Puce

- Le prototype répond à un sous-ensemble très restreint des objectifs d'Eva-Flo (il me semble)
- Développement de l'outil `arenaireplot` :
  - Développement depuis 1 an et demi
  - Certaines choses marchent bien et arrivent à maturité
  - On a commis des fautes de conception difficiles à corriger a posteriori
- Le prototype se sert d'outils externes
  - Que faire quand l'outil externe ne marche pas ?
  - Communication pénible par fichiers ou arguments en ligne de commande
- Il faudra analyser le prototype pour s'abstraire de ses limites

Communication pénible (?)

Concernant la communication pénible entre outils...

Communication pénible (?)

Concernant la communication pénible entre outils...

```
1  oldFullParen = fullparentheses = ?;
2  fullparentheses = on!;
3  if (w==1) then {
4  bashexecute("./myUserFunc.sh \""@f@"\" > ./userFunc.log");
5  } else {
6  bashexecute("./myUserFunc.sh \""@(simplifysafe(1/w))@"\" > ./userFunc.log");
7  };
8  fullparentheses = oldFullParen!;
9  pointsString="";
10 for i in pointsList do pointsString=pointsString@i@" ";
11 monomialsString="";
12 for i in listeMonomials do monomialsString=monomialsString@i@" ";
13 if (w==1) then {
14 bashexecute("./vanderCoeffsSparseAbsErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
15 } else {
16 bashexecute("./vanderCoeffsSparseRelErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
17 };
18 bashexecute("./extractCoeffs.sh");
19 read "bounds.atls";
```

Communication pénible (?)

Concernant la communication pénible entre outils...

```
1  oldFullParen = fullparentheses = ?;
2  fullparentheses = on!;
3  if (w==1) then {
4  bashexecute("./myUserFunc.sh \""@f@"\" > ./userFunc.log");
5  } else {
6  bashexecute("./myUserFunc.sh \""@(simplifysafe(1/w))@"\" > ./userFunc.log");
7  };
8  fullparentheses = oldFullParen!;
9  pointsString="";
10 for i in pointsList do pointsString=pointsString@i@" ";
11 monomialsString="";
12 for i in listeMonomials do monomialsString=monomialsString@i@" ";
13 if (w==1) then {
14 bashexecute("./vanderCoeffsSparseAbsErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
15 } else {
16 bashexecute("./vanderCoeffsSparseRelErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
17 };
18 bashexecute("./extractCoeffs.sh");
19 read "bounds.atls";
```

En fait, c'est juste un appel à un outil écrit dans la même équipe...

Communication pénible (?)

Concernant la communication pénible entre outils...

```
1  oldFullParen = fullparentheses = ?;
2  fullparentheses = on!;
3  if (w==1) then {
4  bashexecute("./myUserFunc.sh \""@f@"\" > ./userFunc.log");
5  } else {
6  bashexecute("./myUserFunc.sh \""@(simplifysafe(1/w))@"\" > ./userFunc.log");
7  };
8  fullparentheses = oldFullParen!;
9  pointsString="";
10 for i in pointsList do pointsString=pointsString@i@" ";
11 monomialsString="";
12 for i in listeMonomials do monomialsString=monomialsString@i@" ";
13 if (w==1) then {
14 bashexecute("./vanderCoeffsSparseAbsErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
15 } else {
16 bashexecute("./vanderCoeffsSparseRelErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
17 };
18 bashexecute("./extractCoeffs.sh");
19 read "bounds.atls";
```

En fait, c'est juste un appel à un outil écrit dans la même équipe...

Et chaque outil nécessite son interfaçage propre à lui...

Communication pénible (?)

Concernant la communication pénible entre outils...

```
1  oldFullParen = fullparentheses = ?;
2  fullparentheses = on!;
3  if (w==1) then {
4  bashexecute("./myUserFunc.sh \""@f@"\" > ./userFunc.log");
5  } else {
6  bashexecute("./myUserFunc.sh \""@(simplifysafe(1/w))@"\" > ./userFunc.log");
7  };
8  fullparentheses = oldFullParen!;
9  pointsString="";
10 for i in pointsList do pointsString=pointsString@i@" ";
11 monomialsString="";
12 for i in listeMonomials do monomialsString=monomialsString@i@" ";
13 if (w==1) then {
14 bashexecute("./vanderCoeffsSparseAbsErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
15 } else {
16 bashexecute("./vanderCoeffsSparseRelErrList-01 "@(prec=?)@" "@err@" "@(length(
       pointsList))@" "@pointsString@(length(listeMonomials))@" "@monomialsString@
       " > resultats.log 2> ./resultsStderrVanderCoeff");
17 };
18 bashexecute("./extractCoeffs.sh");
19 read "bounds.atls";
```

En fait, c'est juste un appel à un outil écrit dans la même équipe...

Et chaque outil nécessite son interfaçage propre à lui...

Il y a ceux qui ne sont pas d'accord avec ce slide ; discutons...

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples
  - Entrée/ sortie pour expressions à une variable, types complexes de l'outils non gérés
  - XML façon MathML-content, pas d'annotations
  - Code XSLT pour la transformation en MathML-presentation (affichage)
  - Entrée/ sortie involutive, aucun autre outil ne supporte l'entrée ou la sortie

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples
  - Entrée/ sortie pour expressions à une variable, types complexes de l'outils non gérés
  - XML façon MathML-content, pas d'annotations
  - Code XSLT pour la transformation en MathML-presentation (affichage)
  - Entrée/ sortie involutive, aucun autre outil ne supporte l'entrée ou la sortie
- XML inutilisable pour l'instant manque de "partenaires"

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples
  - Entrée/ sortie pour expressions à une variable, types complexes de l'outils non gérés
  - XML façon MathML-content, pas d'annotations
  - Code XSLT pour la transformation en MathML-presentation (affichage)
  - Entrée/ sortie involutive, aucun autre outil ne supporte l'entrée ou la sortie
- XML inutilisable pour l'instant manque de "partenaires"
- Syntaxe `arenaireplot` enfin stable, communication possible

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples
  - Entrée/ sortie pour expressions à une variable, types complexes de l'outils non gérés
  - XML façon MathML-content, pas d'annotations
  - Code XSLT pour la transformation en MathML-presentation (affichage)
  - Entrée/ sortie involutive, aucun autre outil ne supporte l'entrée ou la sortie
- XML inutilisable pour l'instant manque de "partenaires"
- Syntaxe `arenaireplot` enfin stable, communication possible
- Conclusions...

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples
  - Entrée/ sortie pour expressions à une variable, types complexes de l'outils non gérés
  - XML façon MathML-content, pas d'annotations
  - Code XSLT pour la transformation en MathML-presentation (affichage)
  - Entrée/ sortie involutive, aucun autre outil ne supporte l'entrée ou la sortie
- XML inutilisable pour l'instant manque de "partenaires"
- Syntaxe `arenaireplot` enfin stable, communication possible
- Conclusions... ... mitigées

Communication...

- Grâce à Nicolas Jourdan, `arenaireplot` peut générer et parser des expressions MathML simples
  - Entrée/ sortie pour expressions à une variable, types complexes de l'outils non gérés
  - XML façon MathML-content, pas d'annotations
  - Code XSLT pour la transformation en MathML-presentation (affichage)
  - Entrée/ sortie involutive, aucun autre outil ne supporte l'entrée ou la sortie
- XML inutilisable pour l'instant manque de "partenaires"
- Syntaxe `arenaireplot` enfin stable, communication possible
- Conclusions... ... mitigées
  - XML intéressant
  - Développement du langage ne doit pas bloquer l'avancement du reste
  - Avec toutes ses questions, on oublie l'algorithmique dans les différents outils

Merci !

Merci de votre attention !

Questions ?