Bitfunctor. Техническое предзадание.

Юзер-процесс.

Единица распространения информации. Тут есть несколько вариантов:

- «Замыкание функции» (funclosure). Включает в себя головную функцию, все функции, используемые головной и далее рекурсивно до арифметики (или CSL?), спецификацию головной функции в виде теорем о ней, доказательство этой теоремы на Coq, и все леммы, используемые для доказательства, рекурсивно до Coq Standard Library (CSL), а также функции, используемые этими леммами, рекурсивно до арифметики (или CSL?). + Типы и индуктивные отношения.
- «Замыкание теоремы (спецификации)» (speclosure). Включает в себя головную теорему, доказательство этой теоремы, и все леммы, используемые для доказательства, рекурсивно до Coq Standard Library (CSL), а также функции, используемые этими леммами, рекурсивно до арифметики (или CSL?). + Типы и индуктивные отношения.
- «Замыкание библиотеки функций» (funclib). Замыкание набора функций, аналогично funclosure.
- «Теория» (theory). Замыкание набора теорем, аналогично speclosure.
- Файлы без определенного знания о них??

Способ распространения информации.

- Funclosure распространяется в виде *.v файлов, *.vo файлов и *.ml файлов
- Speclosure – аналогично, но без ml файлов.

Способ хранения информации.

Предполагается три варианта хранения данных:

- Централизованный. Хостирование и структуризацию данных полагаем на некоторый сервер(а), которые находятся под нашим контролем. Информация доступна 7/24, все reliability – наше;
- Децентрализованный. Полностью распределенная p2p сеть, каждый пир может быть и сидером и личером по желанию.
- Смешанный. Децентрализованный способ, при котором наши сервера выполняют функцию сидера всего, но без какой-либо специальной reliability.

Представляется, что в начале будет смешанный способ, с последующим переходом к одному из первых, в зависимости от того как все пойдет. Учитывая специфику проекта, децентрализованному способу отдается предпочтение.

Functor Universe. Учитывая нелокальность объектов (их связь с тем, что находится не на стороне текущего пира), имеет смысл поддерживать Functor Universe – глобальный граф всего, что находится в системе. Он обновляется каждый раз, когда в системе происходит изменения. В будущем параметры (сложность) этого графа обеспечивают выпуск в системе funcoin-ов, которые распределяются пока по неизвестному алгоритму между участниками. Граф имеет следующую структуру:

- Вершинами (объектами) являются теоремы/леммы, типы/индуктивные отношения
- Дуги (стрелки) бывают такими:

	Функции/Фикспоинты	**Теоремы/Леммы/Именованные аксиомы**	**Типы/Индуктивные отношения**
Функции/Фикспоинты	Содержат(ся) вызов (для фикспоинтов никогда не пусто)	Используются в доказательстве, используются в заголовке	Используют в сигнатуре, используют в теле(?)
Теоремы/Леммы/ Именованные аксиомы	Используют в доказательстве, используют в заголовке	Использут(ся) в доказательстве, использут(ся) в заголовке (только для аксиом)	Используют в доказательстве, используют в заголовке
Типы/Индуктивные отношения	Используются в сигнатуре, используются в теле(?)	Используются в доказательстве, используются в заголовке	Используют(ся) при определении

В связи с этим необходимо определять тождественность объектов, т.е. присваивать им одинаковый хеш, если они математически одинаковы. Это сложная задача. При добавлении любого нового объекта (funclosure или speclosure) происходит его идентификация, хеширование и соответствующее обновление Functor Universe.

Способ адресации (*.functor file scheme). Для централизованного хранения особо сложной адресации, равно как и самих functor файлов может и не быть. Поэтому рассмотрим далее децентрализованный способ. Помимо разной полезной и не очень полезной информации, functor файл должен определенным образом связывать объекты друг с другом. При аплоаде генерация functor файла происходит на стороне трекера (после обновления Functor Universe) и затем отдается сиду в качестве метки и метаинструкции для предстоящего аплоада. При даунлоаде генерация functor файла также происходит на стороне трекера (на основе Functor Universe) и только потом он доступен для скачивания клиентом. Генерация functor файла происходит посредством выбора центрального объекта (для funclosure и speclosure соответственно) и последовательным движением по стрелкам к следующим объектам (только по стрелкам «co»[1]), пока есть стрелки. Данная совокупность определит инструкцию для даунлоада. В принципе можно не хранить FU только на стороне трекера, а хранить его на всех пирах (это обеспечит еще большую децентрализованность и возможность расчета functor файла локально на стороне пира – только для даунлоада, поскольку при аплоаде необходимо будет еще установить соответствие хешей из предлагаемого closure с хранимыми в FU).

[1]Пусть стрелка «co» означает тот факт, что для текущего объекта необходим следующий, а стрелка «contra» - наоборот

Типы юзеров.

Типы юзеров и их взаимодействие **Кто , кому ->**	**Программист**	**Математик**	**Заказчик/владелец кода**	
Программист	Помощь в создании кода	Программирование по выданной спецификации, помощь в формулировках засчет рассказа об используемых функциях, удобных для программирования	Код по сформулировнному ранее заданию (математически)	
Математик	Создание спецификации для существующего кода, формулировка задания (спецификации) на программирование, передаа алгоритма для последуюещего программирования, доказательство уже написанного кода	Помощь в спецификации, в доказательстве	Формулировка спецификации по заданию и доказательства при имеющемся коде	
Заказчик/владелец кода	Ничего напрямую, только через математика	Формулировка своей задачи, для последующего создания спецификации	Помощь в формулировке задачи	