**Как «Forward Proposer» генерирует синтетические теоремы**

*(пересказ статьи “Training a First‑Order Theorem Prover from Synthetic Data” на русском)*

1. **Отправная точка — аксиомы предметной области**  
   Для каждой из десяти выбранных областей TPTP (теория полей, геометрия, теория групп и т. д.) берут набор аксиом в клаузальной форме (CNF).
2. **Линейная резолюция: строим «цепочку» выводов**
   * Пусть C₀ … C\_N — последовательность клауз.
   * C₀ выбирают из аксиом.
   * На каждом шаге *t* = 1…N текущую клаузу C\_{t‑1} **обязательно** используют в следующей резолюции или факторизации, получая C\_t.
   * Так как новая клауза всегда участвует в следующем выводе, дерево вывода вырождается в линию (linear resolution), но при этом остаётся полнотой по Ли.
3. **Стохастический выбор вывода с «температурой»**  
   Простая равномерная выборка быстро порождает гигантские клаузулы. Поэтому каждый возможный вывод оценивают по размеру результирующей клаузулы (числу символов) и выбирают через soft‑max

P(i)=exp⁡(−∣Ci∣/T)∑jexp⁡(−∣Cj∣/T), P(i)=\frac{\exp(-|C\_i|/T)}{\sum\_j\exp(-|C\_j|/T)},P(i)=∑j​exp(−∣Cj​∣/T)exp(−∣Ci​∣/T)​,

где **T** — температура: низкая T усиливает предпочтение к компактным формулам, высокая T даёт больше разнообразия.

1. **Через N шагов получаем гипотезу**  
   После ровно **N** резолюций итоговая клауза C\_N объявляется *конъюнктурой* задачи. Поскольку она выведена из аксиом, импликация

Аксиомы  ⊢  CN \text{Аксиомы} \;\vdash\; C\_NАксиомы⊢CN​

гарантированно истинна.

1. **Подбор параметров N и T по трём критериям**  
   Для каждой области проводят сеточный поиск (N, T) и генерируют 1 млн теорем; затем выбирают сочетание, которое:
   * максимально затрудняет доказательство для E‑prover (сложность),
   * удерживает средний размер клаузы ≤ 64 символов (компактность),
   * обеспечивает ≥ 500 000 уникальных теорем из миллиона (разнообразие).

Пример: для GEO8 взяли N = 10, T = 8.

1. **Поток данных для обучения**  
   С выбранными (N, T) «Forward Proposer» генерирует десятки миллионов теорем, которые служат опытом для нейронного резолюционного доказателя.

**Почему подход работает**

* **Звукoвость по определению** — каждая синтетическая теорема уже доказана построением.
* **Управляемый уровень сложности** — N регулирует глубину вывода, T — сложность формул.
* **Широкий охват домена** — используются только символы аксиом, но пространство теорем гораздо шире, чем в TPTP.

**Ограничения, отмеченные авторами**

* Генерируются только теоремы, сводимые к **одной клаузе** (без ∃‑кванторов и новых символов); это покрывает ≈ 78 % задач TPTP.
* Могут появляться «скучные» или малоинформативные утверждения; нужна мера *интересности*.
* Линейная резолюция не порождает ветвящихся деревьев доказательств.

Если хотите поэкспериментировать:

* реализуйте простой цикл на 10 шагов с soft‑max выбором вывода;
* измеряйте сложность своим доказателем и подбирайте N/T;
* чтобы покрыть ∃‑теоремы, разрешите выводить не одну, а набор клауз или добавляйте свежие сколем‑константы.

Готов предоставить пример кода, идеи расширения на равенство или советы по интеграции с вашим исследованием разрешения и ML — дайте знать!

create\_examples.py

def all\_resolvents(C1, C2):

Ниже приведён один из возможных вариантов реализации функции forward\_propose, которая создаёт линейную цепочку клауз из исходных аксиом. Эта функция выбирает случайную стартовую клаузу (C₀), а затем на каждом шаге пытается разрешить последнюю полученную клаузу с каждой из уже известных, используя метод all\_resolvents (мы пока его оставляем как внешнюю зависимость – его вы реализуете позже). Результат выбирается через softmax‑взвешивание, где более короткие клаузы получают больший вес.

**Пояснения:**

1. **Инициализация**  
   Функция берёт все аксиомы (role 'axiom') из списка, полученного от parse\_tptp\_clauses, и выбирает случайную из них для старта цепочки.
2. **Цикл по шагам (t от 0 до N-1)**  
   На каждом шаге выбираем резолвенты для последней клаузы (chain[-1]) с каждой из известных клауз. Все найденные кандидаты собираются в список.
3. **Выбор кандидата**  
   Если список кандидатов не пуст, то по softmax‑взвешиванию (с весами, обратными размеру клаузы) выбирается одна резолвента, которая добавляется как новая клауза в цепочку и в список известных клауз.
4. **Возврат результата**  
   После завершения всех шагов функция возвращает последнюю клаузу, которая станет заключительной теоремой (conjecture) для дальнейшей генерации задачи.